Journal Title 0 (0) 1
Submitted to IOS Press

Generating Time-Based Label Refinements to Discover More Precise Process Models

Niek Tax [a,*], Emin Alasgarov [b], Natalia Sidorova [a], Reinder Haakma [c], and Wil M.P. van der Aalst [a]
[a] *Eindhoven University of Technology, Eindhoven, The Netherlands*
E-mails: n.tax@tue.nl, n.sidorova@tue.nl, w.m.p.v.d.aalst@tue.nl
[b] *Bol.com, Utrecht, The Netherlands*
E-mail: ealasgarov@bol.com
[c] *Philips Research, Eindhoven, The Netherlands*
E-mail: r.haakma@philips.com

Abstract. Process mining is a research field focused on the analysis of event data with the aim of extracting insights related to dynamic behavior. Applying process mining techniques on data from smart home environments has the potential to provide valuable insights into (un)healthy habits and to contribute to ambient assisted living solutions. Finding the right event labels to enable the application of process mining techniques is however far from trivial, as simply using the triggering sensor as the label for sensor events results in uninformative models that allow for too much behavior (i.e., the models are overgeneralizing). Refinements of sensor level event labels suggested by domain experts have been shown to enable discovery of more precise and insightful process models. However, there exists no automated approach to generate refinements of event labels in the context of process mining. In this paper we propose a framework for the automated generation of label refinements based on the time attribute of events, allowing us to distinguish behaviourally different instances of the same event type based on their time attribute. We show on a case study with real-life smart home event data that using automatically generated refined labels in process discovery, we can find more specific, and therefore more insightful, process models. We observe that one label refinement could have an effect on the usefulness of other label refinements when used together. Therefore, we explore four strategies to generate useful combinations of multiple label refinements and evaluate those on three real-life smart home event logs.

Keywords: Knowledge discovery for smart home environments, Circular statistics, Process mining

1. Introduction

Process mining is a fast growing discipline that combines knowledge and techniques from data mining, process modeling, and process model analysis [1]. Process mining techniques analyze events that are logged during process execution. Today, such event logs are readily available and contain information on what was done, by whom, for whom, where, when, etc. Events can be grouped into cases (process instances), e.g., per patient for a hospital log, or per insurance claim for an insurance company. *Process discovery* plays an important role in process mining, focusing on extracting interpretable models of processes from event logs. One of the attributes of the events is usually used as its label, and its values become transition/activity labels in the process models generated by process discovery algorithms.

The scope of process mining has broadened in recent years from business process management to other application domains, one of them is the analysis of events of human behavior with data originating from sensors in smart home environments [2–7]. Table 1 shows an example of such an event log. Events in the event log are generated by, e.g., motion sensors placed in the house, power sensors placed on appliances, open/close sensors placed on closets and cabinets, etc. Particularly challenging in applying process

*Corresponding author. E-mail: n.tax@tue.nl.

mining in this application domain is the extraction of meaningful event labels that allow for the discovery of insightful process models. Simply using the sensor that generates an event (the *sensor* column in Table 1) as event label is shown to produce non-informative process models that overgeneralize the event log and allow for too much behavior [3]. Abstracting sensor-level events into events at the level of human activity (e.g., *eating*, *sleeping, etc.*) using activity recognition techniques helps to discover more behaviorally constrained and more insightful process models [4, 8]. However, the applicability of this approach relies on the availability of a reliable diary of human behavior at the activity level, which is often expensive or sometimes even impossible to obtain.

Existing approaches that aim at mining temporal relations from smart home environment data [9–14] do not support the rich set of temporal ordering relations that are found in the process models [15], which amongst others include sequential ordering, (exclusive) choice, parallel execution, and loops.

In our earlier work [3], we showed that better process models can be discovered by taking the name of the sensor that generated the event as a starting point for the event label and then refining these labels using information on the time within the day at which the event occurred. The refinements used in [3] were based on domain knowledge, and not identified automatically from the data. In this paper, we aim at the automatic generation of semantically interpretable label refinements that can be explained to the user, by basing label refinements on data attributes of events. We explore methods to bring parts of the timestamp information to the event label in an intelligent and fully automated way, with the end goal of discovering behaviorally more precise and therefore more insightful process models. Initial work on generating label refinements based on timestamp information was started in [16]. Here, we extend the work started in [16] in two ways. First, we propose strategies to select a set of time-based label refinements from candidate time-based label refinements. Furthermore, add an evaluation of the technique in the form of a case study on a real-life smart home dataset.

We start by introducing basic concepts and notations used in this paper in Section 2. In Section 3, we introduce a framework for the generation of event labels refinements based on the time attribute. In Section 4, we apply this framework on a real-life smart home dataset and show the effect of the refined event labels on process discovery. In Section 5, we address the case of applying multiple label refinements together. We continue by describing related work in Section 6 and conclude in Section 7.

2. Background

In this section, we introduce basic notions related to event logs and relabeling functions for traces and then define the notions of refinements and abstractions. We also introduce some Petri net basics.

We use the usual sequence definition, and denote a sequence by listing its elements, e.g., we write $\langle a_1, a_2, \ldots, a_n \rangle$ for a (finite) sequence $s : \{1, \ldots, n\} \rightarrow A$ of elements from some alphabet A, where $s(i) = a_i$ for any $i \in \{1, \ldots, n\}$; $|s|$ denotes the length of sequence s; $s_1 s_2$ denotes the concatenation of sequences s_1 and s_2. A *language* $\mathfrak{L}$ over an alphabet A is a set of sequences over A.

An event is the most elementary element of an event log. Let $\mathcal{I}$ be a set of event identifiers, $\mathcal{T}$ be the time domain, and $\mathcal{A}_1 \times \cdots \times \mathcal{A}_n$ be an attribute domain consisting of n attributes (e.g., resource, activity name, cost, etc.). An event is a tuple $e = (i, a_t, a_1, \ldots, a_n)$, with $i \in \mathcal{I}$, $a_t \in \mathcal{T}$, and $(a_1, \ldots, a_n) \in \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$. The *event label* of an event is the attribute set $(a_1, \ldots, a_n)$. Functions $id(e)$, $label(e)$, and $time(e)$ respectively return the id, the event label and the timestamp of event e. $\mathcal{E} = \mathcal{I} \times \mathcal{T} \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$ is a universe of events over $\mathcal{A}_1, \ldots, \mathcal{A}_n$. The rows of Table 1 are events from an event universe over the event attributes *address*, *sensor*, and *sensor value*.

Events are often considered in the context of other events. We call $E \subseteq \mathcal{E}$ an *event set* if E does not contain multiple events with the same event identifier. The events in Table 1 together form an event set. A *trace* σ is a finite sequence formed by the events from an event set $E \subseteq \mathcal{E}$ that respects the time ordering of events, i.e., for all $k, m \in \mathbb{N}$, $1 \leqslant k < m \leqslant |E|$, we have: $time(\sigma(k)) \leqslant time(\sigma(m))$. We define the *universe of traces* over event universe $\mathcal{E}$, denoted $\Sigma(\mathcal{E})$, as the set of all possible traces over $\mathcal{E}$. We omit $\mathcal{E}$ in $\Sigma(\mathcal{E})$ and use the shorter notation Σ when the event universe is clear from the context.

Often it is useful to partition an event set into smaller sets in which events belong together according to some criterion. We might for example be interested in discovering the typical behavior within a household over the course of a day. In order to do so, we can group together events with the same *address* and the same day-part of the *timestamp*, as indicated by the horizon-

Table 1
An example of an event log from a smart home environment.

Id	Timestamp	Address	Sensor	Sensor value
1	03/11/2015 04:59:54	Mountain Rd. 7	Motion sensor - Bedroom	1
2	03/11/2015 06:04:36	Mountain Rd. 7	Motion sensor - Bedroom	1
3	03/11/2015 08:45:12	Mountain Rd. 7	Motion sensor - Living room	1
4	03/11/2015 09:10:10	Mountain Rd. 7	Motion sensor - Kitchen	1
5	03/11/2015 09:12:01	Mountain Rd. 7	Power sensor - Water cooker	1200
6	03/11/2015 09:15:45	Mountain Rd. 7	Power sensor - Water cooker	0
...	03/11/2015 ...	Mountain Rd. 7	...	...
7	03/12/2015 01:01:23	Mountain Rd. 7	Motion sensor - Bedroom	1
8	03/12/2015 03:13:14	Mountain Rd. 7	Motion sensor - Bedroom	1
9	03/12/2015 07:24:57	Mountain Rd. 7	Motion sensor - Bedroom	1
10	03/12/2015 08:34:02	Mountain Rd. 7	Motion sensor - Bedroom	1
11	03/12/2015 09:12:00	Mountain Rd. 7	Motion sensor - Living room	1
...	03/12/2015 ...	Mountain Rd. 7	...	...
...	...	...	...	...

tal lines in Table 1. For each of these event sets, we can construct a trace; timestamps define the ordering of events within the trace. For events of a trace having the same timestamps, an arbitrary ordering can be chosen within a trace.

An *event partitioning function* is a function $ep : \mathcal{E} \rightarrow T_{id}$ that defines the partitioning of an arbitrary set of events $E \subseteq \mathcal{E}$ from a given event universe $\mathcal{E}$ into event sets $E_1, \ldots, E_j, \ldots$ where each E_j is the maximal subset of E such that for any $e_1, e_2 \in E_j$, $ep(e_1) = ep(e_2)$; the value of ep shared by all the elements of E_j defines the value of the *trace attribute* T_{id}. Note that multidimensional trace attributes are also possible, i.e., a combination of the name of the person performing the event activity and the date of the event, so that every trace contains activities of one person during one day. The event sets obtained by applying an event partitioning can be transformed into traces (respecting the time ordering of events).

An event log L is a finite set of traces $L \subseteq \Sigma(\mathcal{E})$ such that $\forall \sigma \in L : \forall e_1, e_2 \in \sigma : ep(e_1) = ep(e_2)$. $A_L \subseteq \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$ denotes the *alphabet of event labels* that occur in log L. The traces of a log are often transformed before doing further analysis: very detailed but not necessarily informative event descriptions are transformed into some *coarse-grained* and *interpretable* labels. For the labels of the log in Table 1, the sensor values could be abstracted to *on* and *off*, or labels can be redefined to a subset of the event attributes, e.g., leaving the sensor values out completely.

After this relabeling step, some traces of the log can become identically labeled (the event id's would still be different). The information about the number of occurrences of a sequence of labels in an event log is highly relevant for process mining, since it allows process discovery algorithms to differentiate between the mainstream behavior of a process (i.e., frequently occurring behavioral patterns) and the exceptional behavior.

Let $\mathcal{E}_1, \mathcal{E}_2$ be event universes. A function $l : \mathcal{E}_1 \rightarrow \mathcal{E}_2$ is an *event relabeling function* when it satisfies $id(e) = id(l(e))$ and $time(e) = time(l(e))$ for all events $e \in \mathcal{E}_1$. A relabeling function can be used to obtain more useful event labels than the full set of event attribute values, by lifting those elements of the attribute space to the label that result in strong ordering relations in the resulting log. We lift l to event logs. Let $\mathcal{E}, \mathcal{E}_1, \mathcal{E}_2$ be event universes with $\mathcal{E}, \mathcal{E}_1, \mathcal{E}_2$ being pairwise different. Let $l_1 : \mathcal{E} \rightarrow \mathcal{E}_1$ and $l_2 : \mathcal{E} \rightarrow \mathcal{E}_2$ be event relabeling functions. Relabeling function l_1 is a *refinement* of relabeling function l_2, denoted by $l_1 \preceq l_2$, iff $\forall_{e_1,e_2 \in \mathcal{E}} : label(l_1(e_1)) = label(l_1(e_2)) \implies label(l_2(e_1)) = label(l_2(e_2))$; l_2 is then called an *abstraction* of l_1.

The goal of process discovery is to discover a process model that represents the behavior seen in an event log. The activities/transitions in this discovered process model describe allowed orderings over the labels of the events in the event logs. A frequently used process modeling notation in the process mining field is the Petri net notation [17]. Petri nets are directed bi-

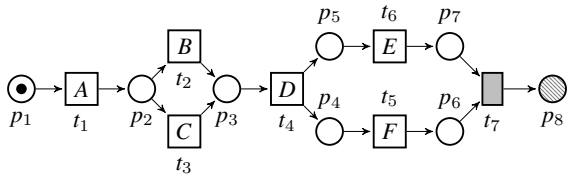


Fig. 1. An example Petri net.

partite graphs consisting of transitions and places, connected by arcs. Transitions represent activities, while places represent the enabling conditions of transitions. Labels are assigned to transitions to indicate the type of activity that they model. A special label τ is used to represent invisible transitions, which are only used for routing purposes and not recorded in the log.

A *labeled Petri net* $N = \langle P, T, F, A_M, \ell \rangle$ 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, A_M is an alphabet of labels representing activities, with $\tau \notin A_M$ being a label representing invisible events, and $\ell : T \rightarrow A_M \cup \{\tau\}$ is a labeling function that assigns a label to each transition. For a node $n \in P \cup T$ we use $\bullet n$ and $n\bullet$ to denote the set of input and output nodes of n, defined as $\bullet n = \{n' \mid (n', n) \in F\}$ and $n\bullet = \{n' \mid (n, n') \in F\}$. An example of a Petri net can be seen in Figure 1, where circles represent places and rectangles represent transitions. Gray transitions having a smaller width represent invisible, or τ, transitions.

A state of a Petri net is defined by its *marking* $M \in \mathbb{N}^P$ being a multiset of places. A marking is graphically denoted by putting $M(p)$ tokens on each place $p \in P$. A pair (N, M) is called a *marked Petri net*. 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 a transition fires, one token is removed from each input place of t and one token is added to each output place of t, leading to a new marking. An *accepting Petri net* is a 3-tuple (N, M_i, M_f) with N a labeled Petri net, M_i an initial marking, and M_f a final marking. Visually, places that belong to the initial marking contain a token (e.g., p_1 in Figure 1), and places that belong to the final marking are depicted as ◍. Many process modeling notations, including accepting Petri nets, have formal executional semantics and a model defines a *language of accepting traces* $\mathfrak{L}$. The language of a Petri net consists of all sequences of activities that have a firing sequence through the Petri net that starts in the initial marking and ends in the final marking. For the Petri net in Figure 1, the language of accepting traces is $\{\langle A, B, D, E, F\rangle, \langle A, B, D, F, E\rangle, \langle A, C, D, E, F\rangle, \langle A, C, D, F, E\rangle\}$. In words: the process starts with activity A, followed by a choice between activity B and C, followed by activity D, finally followed by activity E and F in parallel (i.e., they can occur in any order). We refer the reader to [17] for a more thorough introduction of Petri nets.

For an event log L and a process model M we say that L is *fitting* on process model M if $L \subseteq \mathfrak{L}(M)$. *Precision* is related to the behavior that is allowed by a process model M that was not observed in the event log L, i.e., $\mathfrak{L}(M) \backslash L$. The aim of process discovery is to discover a process model based on and event log L that has both high *fitness* (i.e., it allows for the behavior seen in the log) and high *precision* (i.e., it does not allow for too much behavior that was not seen in the log). Many process discovery algorithms have been proposed throughout the years, including techniques based on Integer Linear Programming and the theory of regions [18], Inductive Logic Programming [19], maximal pattern mining [20], or based on heuristic techniques [21, 22]. We refer the reader to [1] for a thorough introduction of several process discovery techniques.

In process discovery tasks on event logs from the business process management domain, events are often simply relabeled to the value of an *activity name* attribute, which stores a generally understood name for the event (e.g., *receive loan application*, or *decide on building permit application*). However, event logs from the smart home environment domain generally do not contain a single attribute such that relabeling on that attribute enables the discovery of insightful process models [3]. In this paper we explore a strategy to refine an event label that is based on the name of the sensor in a smart home with information about the time in the day at which the sensor was triggered.

3. A Framework for Time-based Label Refinements

In this section, we describe a framework to generate an event label that contains partial information about the event timestamp, in order to make the event labels more specific while preserving interpretability. Note that by bringing time-in-the-day information to the event label we aim at uncovering daily routines of the person under study. We take a clustering-based approach by identifying dense areas in time-space for

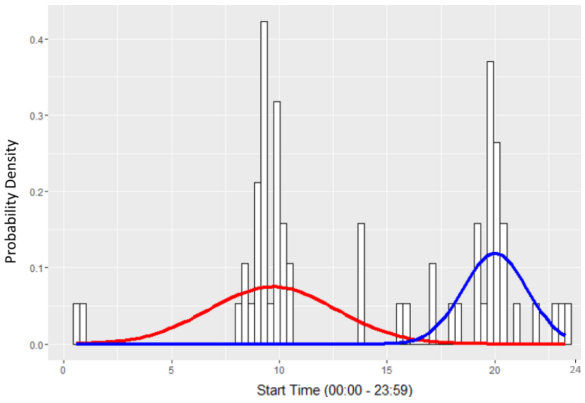


Fig. 2. The histogram representation and a Gaussian Mixture Model fitted to timestamps values of the plates cupboard sensor in the Van Kasteren [24] dataset.

each label. The time part of the timestamps consists of values between *00:00:00* and *23:59:59*, equivalent to the timestamp attribute from Table 1 with the day-part of the timestamp removed. This timestamp can be transformed into a real number time representation in the interval $[0, 24)$. We chose to apply soft clustering (also referred to as fuzzy clustering), which has the benefit of assigning to each data point a likelihood of belonging to each cluster. A well-known approach to soft clustering is based on the combination of the Expectation-Maximization (EM) [23] algorithm with mixture models, which are probability distributions consisting of multiple components of the same probability distribution. Each component in the mixture represents one cluster, and the probability of a data point belonging to that cluster is the probability that this cluster generated that data point. The EM algorithm is used to obtain a maximum likelihood estimate of the mixture model parameters, i.e., the parameters of the probability distributions in the mixture.

A well-known type of mixture model is the Gaussian Mixture Model (GMM), where the components in the mixture distributions are normal distributions. The data space of time is, however, non-Euclidean: it has a circular nature, e.g., 23.99 is closer to 0 than to 23. This circular nature of the data space introduces problems for GMMs. Figure 2 illustrates the problem of GMMs in combination with circular data by plotting the timestamps of the bedroom sensor events of the Van Kasteren [24] real-life smart home event log. The GMM fitted to the timestamps of the sensor events consists of two components, one with the mean at 9.05 (in red) and one with a mean at 20 (in blue). The histogram representation of the same data shows that some events occurred just after midnight, which on the clock is closer to 20 than to 9.05. The GMM, however, is unaware of the circularity of the clock, which results in a mixture model that seems inappropriate when visually comparing it with the histogram. The standard deviation of the mixture component with a mean at 9.05 is much higher than one would expect based on the histogram as a result of the mixture model trying to explain the data points that occurred just after midnight. The field of circular statistics (also referred to as directional statistics), concerns the analysis of such circular data spaces (cf. [25]). In this paper, we use a mixture of *von Mises* distributions to capture the daily patterns.

Here, we introduce a framework for generating refinements of event labels based on time attributes using techniques from the field of circular statistics. This framework consists of three stages to apply to the set of timestamps of a sensor:

Data-model pre-fitting stage A known problem with many clustering techniques is that they return clusters even when the data should not be clustered. In this stage, we assess if the events of a certain sensor should be clustered at all, and if so, how many clusters it contains. For sensor types that are assessed to not be clusterable (i.e., the data consists of one cluster), the procedure ends and the succeeding two stages are not executed.

Data-model fitting stage In this stage, we cluster the events of a sensor type by timestamp using a mixture consisting of components that take into account the circularity of the data. The clustering result obtained in the fitting stage is now a candidate label refinement. The label can be refined based on the clustering result by adding the assigned cluster to the label of the event, e.g., *open/close fridge* can be relabeled into three distinct labels *open/close fridge 1*, *open/close fridge 2*, and *open/close fridge 3* in case the timestamps of the fridge where clustered into three clusters.

Data-model post-fitting stage In this stage, the quality of the candidate label refinements is assessed from both a cluster quality perspective and a process model (event ordering statistics) perspective. The label is only refined when the candidate label refinement is 1) based on a clustering that has a sufficiently good fit with the data, and 2) helps to discover a more insightful process model. If the candidate label refinement does not pass one of the two tests, the label refinement candidate will

not be applied (i.e., the label will remain to only consist of the sensor name).

We now proceed with introducing the three stages in detail.

3.1. Data-model pre-fitting stage

This stage consists of three procedures: a test for uniformity, a test for unimodality, and a method to select the number of clusters in the data. If the timestamps of a sensor type are consider to be uniformly distributed or follow a unimodal distribution, the data is considered to not be clusterable, and the sensor type will not be refined. If the timestamps are neither uniformly distributed nor unimodal, then the procedure for the selection of number of clusters will decide on the number of clusters used for clustering.

3.1.1. Uniformity Check

Rao's spacing test [26] tests the uniformity of the timestamps of the events from a sensor around the circular clock. This test is based on the idea that uniform circular data is distributed evenly around the circle, and n observations are separated from each other $\frac{2\pi}{n}$ radii. The null hypothesis is that the data is uniform around the circle.

Given n successive observations $f_1, \ldots, f_n$, either clockwise or counterclockwise, the test statistics U for Rao's Spacing Test is defined as $U = \frac{1}{2}\sum_{i=1}^{n} \mid T_i - \lambda \mid$, where $\lambda = \frac{2\pi}{n}$, $T_i = f_{i+1} - f_i$ for $1 \leqslant i \leqslant n-1$ and $T_n = (2\pi - f_n) + f_1$.

3.1.2. Unimodality Check

Hartigan's dip test [27] tests the null hypothesis that the data follows a unimodal distribution on a circle. When the null hypothesis can be rejected, we know that the distribution of the data is at least bimodal. Hartigan's dip test measures the maximum difference between the empirical distribution function and the unimodal distribution function that minimizes that maximum difference.

3.1.3. Selecting the Number of Mixture Components

The Bayesian Information Criterion (BIC) [28] introduces a penalty for the number of model parameters to the evaluation of a mixture model. Adding a component to a mixture model increases the number of parameters of the mixture with the number of parameters of the distribution of the added component. The likelihood of the data given the model can only increase by adding extra components, adding the BIC penalty results in a trade-off between the number of components and the likelihood of the data given the mixture model. BIC is formally defined as $BIC = -2{*}ln(\hat{L}) + k{*}ln(n)$, where $\hat{L}$ is a maximized value for the data likelihood, n is the sample size, and k is the number of parameters to be estimated. A lower BIC value indicates a better model. We start with one component and iteratively increase the number of components from k to $k+1$ as long as the decrease in BIC is larger than 10, which is shown to be an appropriate threshold in [29].

3.2. Data-model fitting stage

A generic approach to estimate a probability distribution from data that lies on a circle or any other type of manifold (e.g., the torus and sphere) was proposed by Cohen and Welling in [30]. However, their approach estimates the probability distribution on a manifold in a non-parametric manner, and it does not use multiple probability distribution components, making it unsuitable as a basis for clustering.

We cluster events generated by one sensor using a mixture model consisting of components of the von Mises distribution, which is the circular equivalent of the normal distribution. This technique is based on the approach of Banerjee et al. [31] that introduces a clustering method based on a mixture of von Mises-Fisher distribution components, which is a generalization of the 2-dimensional von Mises distribution to n-dimensional spheres. A probability density function for a von Mises distribution with mean direction μ and concentration parameter κ is defined as $pdf(\theta \mid \mu, \kappa) = \frac{1}{2\pi I_0(\kappa)} e^{\kappa\cos(\theta-\mu)}$, where mean μ and data point θ are expressed in radians on the circle, such that $0 \leqslant \theta \leqslant 2\pi$, $0 \leqslant \mu \leqslant 2\pi$, $\kappa \geqslant 0$. I_0 represents the modified Bessel function of order 0, defined as $I_0(k) = \frac{1}{2\pi}\int_0^{2\pi} e^{\kappa\cos(\theta)} d\theta$. As κ approaches 0, the distribution becomes uniform around the circle. As κ increases, the distribution becomes relatively concentrated around the mean μ and the von Mises distribution starts to approximate a normal distribution. We fit a mixture model of von Mises components using the package movMF [32] provided in R, using the number of components found with the BIC procedure of the pre-fitting stage. A candidate label refinement is created based on the clustering result, where the original label based on the sensor type is refined into a new number of distinct labels, each representing one von Mises component, where each event is relabeled according to the von Mises component that has the assigns the highest likelihood to the timestamp of that event.

3.3. Data-model post-fitting stage

This stage consists of two procedures: a statistical test to assess how well the clustering result fits the data, and a test to assess whether the ordering relations in the log become stronger by applying the relabeling function (i.e., whether it becomes more likely to discover a precise process model with process discovery techniques).

3.3.1. Goodness-of-fit test

After fitting a mixture of von Mises distributions to the sensor events, we perform a goodness-of-fit test to check whether the data could have been generated from this distribution. We describe the Watson U^2 statistic [33], a goodness-of-fit assessment based on hypothesis testing. The Watson U^2 statistic measures the discrepancy between the cumulative distribution function $F(\theta)$ and the empirical distribution function $F_n(\theta)$ of some sample θ drawn from some population and is defined as $U^2 = n \int_0^{2\pi} \Big[F_n(\theta) - F(\theta) - \int_0^{2\pi} \big\{F_n(\phi) - F(\phi)\big\} dF(\phi)\Big]^2 dF(\theta)$.

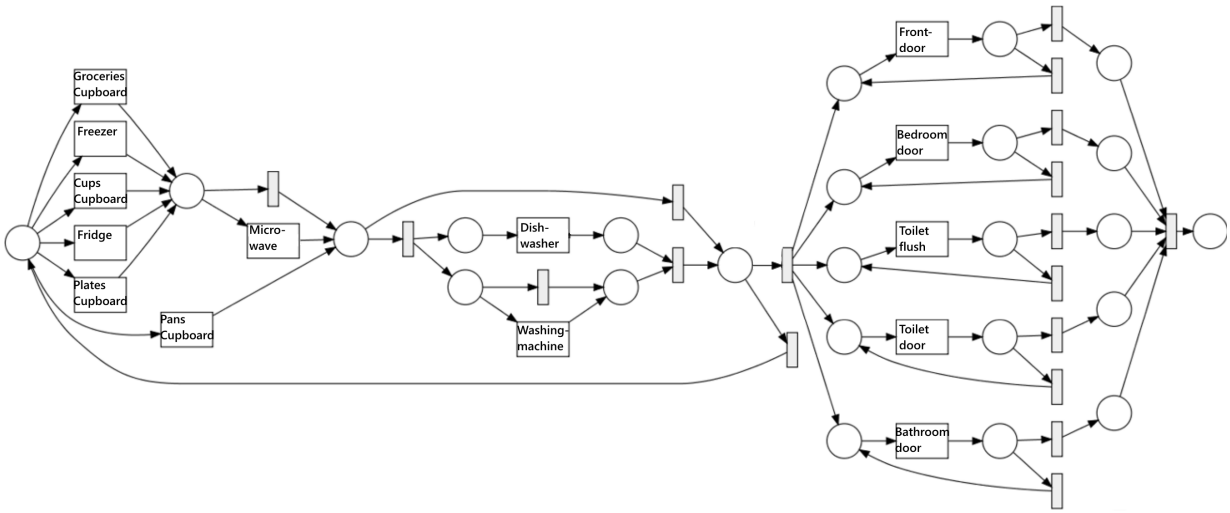
3.3.2. Control flow test

The clustering obtained can be used as a label refinement where we refine the original event label into a new label for each cluster. We assess the quality of this label refinement from a process perspective using the label refinement evaluation method described in [3]. This method tests whether the *log statistics* that are used internally in many process discovery algorithms become significantly more deterministic by applying the label refinement. Hence, we test whether the models become more precise after time-based label refinement. An example of such a log statistic is the *direct successor* statistic: $\#^+_{L,>}(b,c)$ is the number of occurrences of b in the traces of L that are directly followed by c, i.e., in some $\sigma \in L, i \in \{1,\ldots,|\sigma|\}$ we have $label([\sigma(i)]) = b$ and $label([\sigma(i+1)]) = c$, likewise, $\#^-_{L,>}(b,c)$ is the number of occurrences of b which are not directly followed by c. This control-flow test [3] outputs a *p-value* that indicates whether such log statistics of refined activities $a_1, a_2, \ldots$ of some activity a change with statistical significance. When $\#^+_{L,>}(b,c) = \#^-_{L,>}(b,c)$ the entropy of b being directly followed by c is 1 bit, equal to a coin toss. In addition to the p-value, the test returns an *information gain* value, which indicates the ratio of the decrease in the total bits of entropy in the log statistics as a result of applying the label refinement. Information gain can be used as a selection criterion for label refinements when there are multiple sensor types that can be refined according to the three steps of this framework. While the entropy of a single log statistic cannot increase by applying a label refinement, the information gain of a refinement can still be negative when it is not useful, as it increases the number activities in the log and therefore also increases the total number of log statistics.

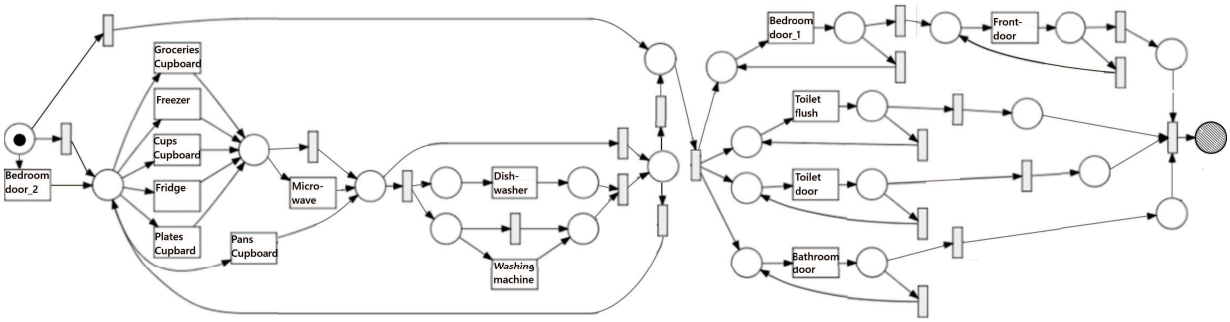
4. Case Study

We apply our time-based label refinements approach to the real-life smart home dataset described in Van Kasteren et al. [24]. The Van Kasteren dataset consists of 1285 events divided over fourteen different sensors. We segment in days from midnight to midnight to define cases. Figure 3a shows the process model discovered on this event log with the Inductive Miner infrequent [34] process discovery algorithm with 20% filtering, which is a state-of-the-art process discovery algorithm that discovers a process model that describes the most frequent 80% of behavior in the log. Note that this process model overgeneralizes, i.e., it allows for too much behavior. At the beginning a (possibly repeated) choice is made between five transitions. At the end of the process, the model allows any sequence over the alphabet of five activities, where each activity occurs at least once.

We illustrate the framework by applying it to the *bedroom door* sensor. Rao's spacing test results in a test statistic of 241.0 with 152.5 being the critical value for significance level 0.01, indicating that we can reject the null hypothesis of a uniformly distributed set of *bedroom door* timestamps. Hartigan's dip test results in a p-value of 3.95×10^{-4}, indicating that we can reject the null hypothesis that there is only one cluster in the *bedroom door* data. Figure 4 shows the BIC values for different numbers of components in the model. The figure indicates that there are two clusters in the data, as this corresponds to the lowest BIC value. Table 2 shows the mean and κ parameters of the two clusters found by optimizing the von Mises mixture model with the EM algorithm. A value of $0 \equiv 2\pi$ radii equals midnight. After applying the von Mises mixture model to the *bedroom door* events and assigning each event to the maximum likelihood cluster we obtain a time range of [3.08-10.44] for cluster 1 and a time range of [17.06-0.88] for cluster 2. The Watson U^2 test results in a test statistic of 0.368 and 0.392 for cluster 1 and 2 respectively with a critical value of 0.141 for a 0.01 significance level, indicating that the data is



(a) Original event data



(b) Relabeled event data

Fig. 3. Process models discovered on the Van Kasteren data with sensor-level labels (a) and refined labels (b) with the Inductive Miner infrequent (20% filtering).

likely to be generated by the two von Mises distributions found. The label refinement evaluation method [3] finds statistically significant differences between the events from the two *bedroom door* clusters with regard to their control-flow relations with other activities in the log for 10 other activities using the significance level of 0.01, indicating that the two clusters are different from a control-flow perspective. Figure 3b shows the process model discovered with the Inductive Miner infrequent with 20% filtering after applying this label refinement to the Van Kasteren event log. The process model still overgeneralizes the overall process, but the label refinement does help to restrict the behavior, as it shows that the evening *bedroom door* events are succeeded by one or more events of type *groceries cupboard*, *freezer*, *cups cupboard*, *fridge*, *plates cupboard*, or *pans cupboard*, while the morning *bedroom door* events are followed by one or more *frontdoor* events. It seems that this person generally goes to the bedroom in-between coming home from work and starting to cook. The loop of the *frontdoor* events could be caused by the person leaving the house in the morning for work, resulting in no logged events until the person comes home again by opening the *frontdoor*. Note that in Figure 3a *bedroom door* and *frontdoor* events can occur an arbitrary number of times in any order. Figure 3a furthermore does not allow for the *bedroom door* to occur before the whole block of kitchen-located events at the beginning of the net. In the process mining field multiple quality criteria exist to express the fit between a process model and an event log. Two of those criteria are *fitness* [35], which measures the degree to which the behavior that is observed in the event log can be replayed on the process model, and *precision* [36], which measures the degree to which the behavior that was never observed in the event log cannot be re-

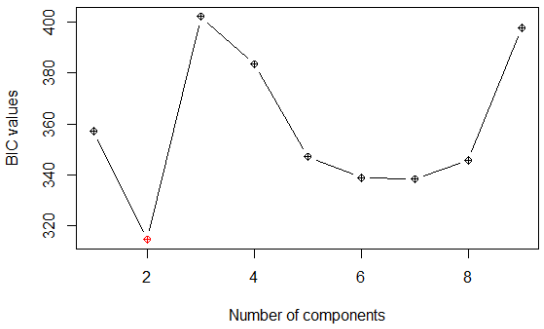


Fig. 4. BIC values for different numbers of components in the mixture model.

Table 2 Estimated parameters for a mixture of von Mises components for bedroom door sensor events.

Cluster	α	μ (radii)	κ
Cluster 1	0.76	2.05	3.85
Cluster 2	0.24	5.94	1.56

played on the process model. Low precision typically indicates an overly general process model, that allows for too much behavior. Typically we aim for process models with both high fitness and precision, therefore one can consider the harmonic mean of the two, often referred to as *F-score*. The *bedroom door* label refinement described above improves the precision of the process model found with the Inductive Miner infrequent (20% filtering) [34] from 0.3577 when applied on the original event log to 0.4447 when applied on the refined event log and improves the F-score from 0.5245 to 0.6156.

The label refinement framework allows for refinement of multiple activities in the same log. For example, label refinements can be applied iteratively. Figure 5 shows the effect of a second label refinement step, where *Plates cupboard* using the same methodology is refined into two labels, representing time ranges [7.98-14.02] and [16.05-0.92] respectively. This refinement shows the additional insight that the evening version of the *Plates cupboard* occurs directly before or after the microwave. Generating multiple label refinements, however, comes with the problem that the control-flow test [3] is sensitive to the order in which label refinements are applied. Because label refinements change the event log, it is possible that after applying some label refinement A, some other label refinement B starts passing the control-flow test that did not pass this test before, or fails the test while it passed before. Additionally, applying one label refinement can change the information gain of applying another label refinement afterwards. For example, when $\#^{+}_{L,>}(b,c) = \#^{-}_{L,>}(b,c)$, i.e., b is followed by c 50% of the time, the entropy of this log statistic is 1, equal to a coin toss. Some label refinement A which refines b into b_1, b_2 where b_1 is always followed by c and b_2 is never followed by c is a good label refinement from an information gain point of view, as it decreases the entropy of the log statistic to zero. Some other label refinement B, which refines c into c_1, c_2 such that all b's are directly followed by c_1's and never by c_2's also leads to information gain. However, applying refinement B after having already applied refinement A, does not lead to any further information gain, since refinement A has already made it deterministic whether or not b is followed by any c. Ineffective label refinements might even harm process discovery, as each refinement decreases the frequencies with which activities are observed, thereby decreasing the amount of evidence for certain control-flow relations.

5. On the Ordering of Label Refinements

As shown in Section 4, the outcome of the control flow test test of a label refinement can depend on whether other label refinements that passed the test of the pre-fitting and post-fitting stages have already been applied. Therefore, in this section, we explore the effect of the ordering of label refinements on real-life event logs. We explore this effect by evaluating four strategies to select a set of label refinements to apply to the event log. Each of the strategies assume the desired number k of label refinements to be given.

All-at-once In this strategy we naively ignore the influence of the interplay between label refinements on the outcome of the control flow test and select top k label refinements in a single step based on their information gain that is calculated using the original event log, to which the other selected label refinements are not applied.

Greedy Search We first apply the best label refinement in terms of information gain, then refine the event log using this label refinement, and then iterate to find the next label refinement calculating the information gain using the refined event log from the previous step.

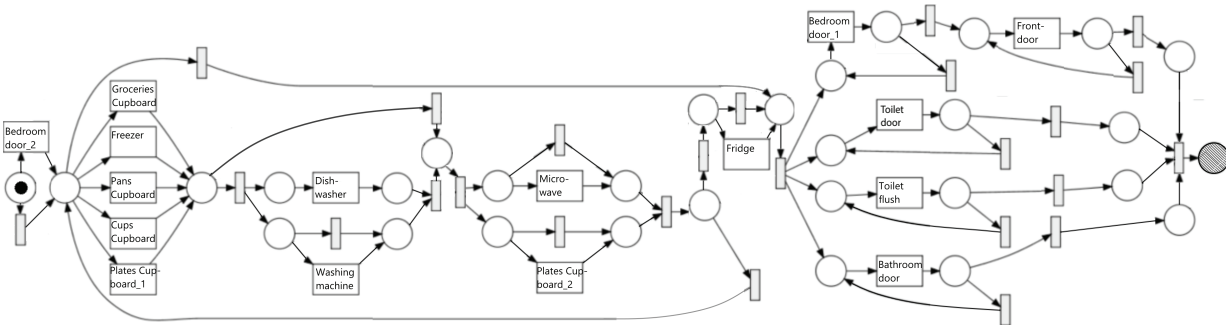


Fig. 5. Inductive Miner infrequent (20% filtering) result after a second label refinement.

Exhaustive Search This strategy exhaustively tries all combinations of label refinements and searches for the label refinement combinations that jointly lead to the largest information gain. While the label refinement combinations that are found with this strategy are optimal in terms of information gain, this strategy can quickly become computationally intractable for event logs that contain many activities.

Beam Search In Beam Search only a predetermined number b (called the beam size) of best partial solutions are kept as candidates, i.e., only the best b combinations in terms of information gain that were found consisting of n label refinements are explored to search for a new set of $n + 1$ label refinements. This is an intermediate strategy in-between greedy and exhaustive search, with greedy search being a beam search with b=1 and exhaustive search being a beam search with b=∞.

We apply these four strategies on three event logs from the human behavior domain and measure the *fitness*, *precision*, and *F-score* of the model discovered with the Inductive Miner infrequent [34] with 20% filtering after each label refinement. The first event log is the Van Kasteren [24] event log which we introduced in Section 4. The other two event logs are two different households of a smart home experiment conducted by MIT [37]. The log *Household A* of the MIT experiment contains 2701 events spread over 16 days, with 26 different sensors. The *Household B* log contains 1962 events spread over 17 days and 20 different sensors.

Figure 6 shows the results. On all three event logs the precision can be improved considerably through label refinements. Note that when applying only one label refinement all four strategies are identical. When refining a second label the four strategies all select the same label refinement on all three logs. Therefore the F-score, fitness, and precision for two refined labels happen to be identical. Figure 6 shows that for the MIT household A data set there are 7 sensor types that can be refined, i.e., they passed the statistical tests of the pre-fitting stage and their obtained clustering passed the goodness-of-fit test. For the MIT household B data set there are 10 activities that can be refined and for the are 8 activities that can be refined for the Van Kasteren data set. However, since the F-score for all strategies drops again after a few label refinements, not all of those label refinements lead to better process models. The four strategies perform very similar in terms of F-score. Exhaustive search outperforms the other strategies for a few refinements on some logs, however, such improvements come with considerable computation times. On the MIT household B log, which has 10 possible label refinements, it takes about 25 minutes on an Intel i7 processor to evaluate all possible combinations of refinements. On logs with even more possible refinements the exhaustive strategy can quickly become computationally infeasible. The all-at-once strategy, which is computationally very fast and only takes milliseconds to compute, shows almost identical performance for MIT household A and Van Kasteren. When making six or more refinements on the MIT household B log, the performance of the all-at-once strategy lags behind the other strategies, indicating that the label refinements that were applied earlier cause the later label refinements to be less effective. However, the optimum in F-score for this log lies at three refinements, therefore the sixth refinement, where a performance difference between non-exhaustive strategies emerges should not be performed with any of the strategies in the first place.

Since the F-score decreases again when applying too many label refinements it is important to have a stopping criterion that prevents refining the event log too much. The dashed line in Figure 6 shows the re-

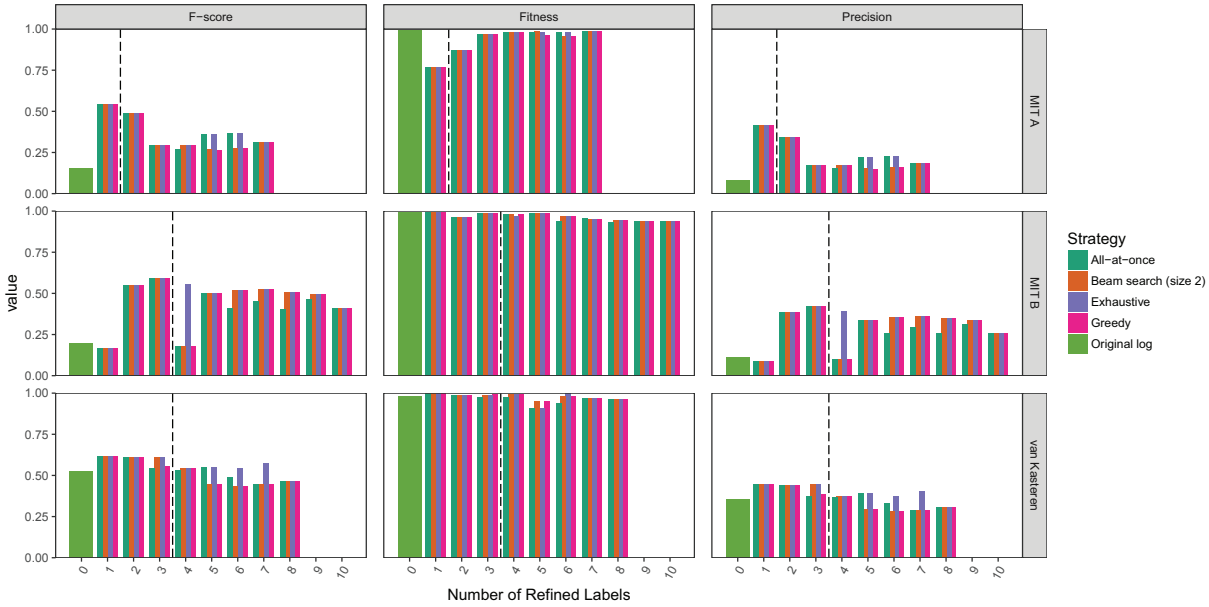


Fig. 6. Fitness, precision, and F-score of the Inductive Miner infrequent (20% filtering) models obtained from the original and refined versions of three event logs.

sults when we only refine a label when the information gain of the refinement is larger than zero. On the MIT households A and B logs this stopping criterion causes all strategies to stop at the best combination of label refinements in F-score, consists respectively of one and three refinements. This indicates that the control flow test [3] provides a useful stopping criterion for label refinements.

All strategies except the exhaustive search strategy suggest as the fourth refinement for MIT B a refinement that decreases the F-score sharply, to increase it again with a fifth refinement. This is caused by an unhelpful refinement being found as the fourth refinement by those strategies, which causes the frequencies to drop below the filtering threshold of the Inductive Miner, leading to a model that is less precise. At the fifth refinement, the follows statistics of other activities drop as well, causing the follows statistics that dropped in the fourth refinement to be relatively higher and above the threshold again. On the Van Kasteren log the optimum in F-score is to make only one refinement, although the F-score after applying the second and third refinement as found by the exhaustive and beam search is almost identical. The all-at-once strategy stops after applying only two refinements while the other strategies apply a third refinement. The best refinement combination found with the all-at-once strategy using the stopping criterion is identical to the refinement combination found with the other strategies, suggesting that in practice the differences between the four approaches are small. On real-life smart home environment event logs the effect that one label refinement influences the control flow test outcome of others is limited.

Figures 7 and 8 shows the process model that are discovered with the Inductive Miner infrequent with 20% filtering respectively from the original MIT household A event log and the event log obtained after applying the optimal combination of label refinements found in the results of Figure 6. Because of the silent transitions, the process model discovered from the original event log allows for almost all orderings over the sensor types. Even though the transition labels in the process model discovered from the refined event log are not readable because of the size, it is clear from the structure of the process model that it is much more behaviorally specific, containing a mix of sequential orderings, parallel blocks, and choices over the sensor types. Especially interesting is the part indicated by the blue dashed ellipse, which contains a parallel block consisting of a *cabinet*, the *oven* and *burner*, and the *dishwasher*, showing a clearly recognizable cooking routine. Furthermore, the part indicated by the red dotted ellipse indicates a sequentially ordered part, consisting of some *door* sensor registering the opening of

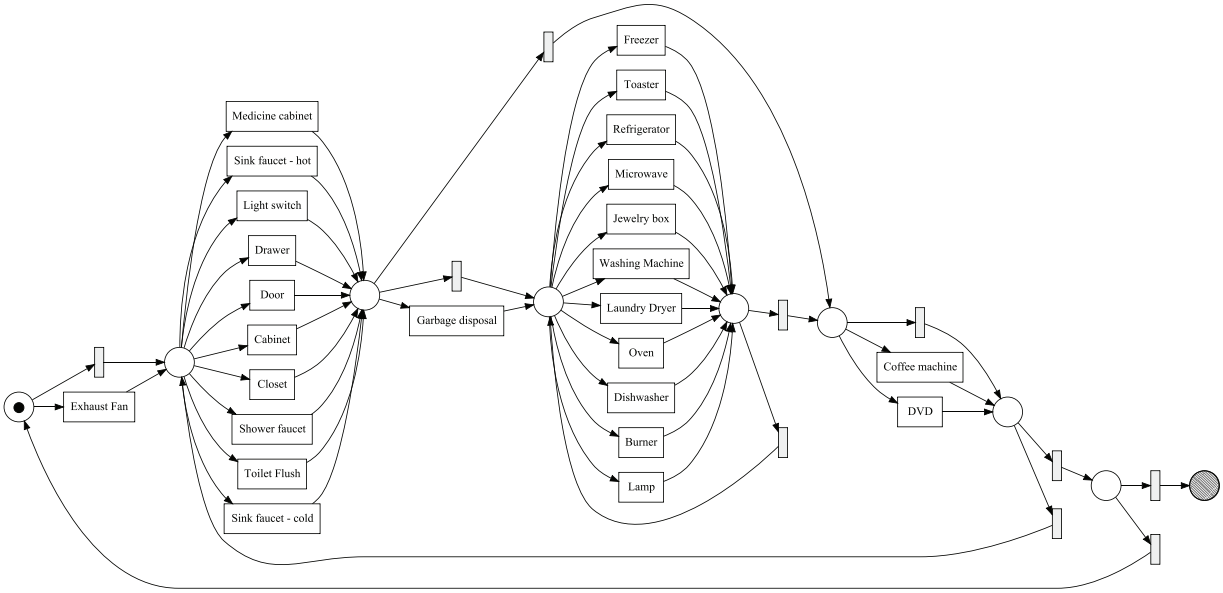


Fig. 7. The Inductive Miner infrequent (20% filtering) process model discovered from the original MIT A event log.

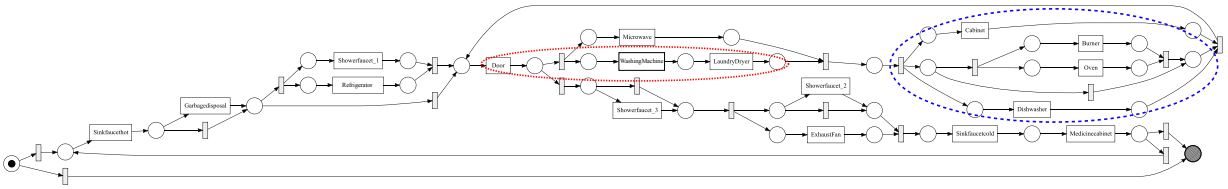


Fig. 8. The Inductive Miner infrequent (20% filtering) process model discovered from the refined MIT A event log.

a door, followed by starting the *washing machine* and then the *laundry dryer*.

The time-based label refinement generation framework as well as the four strategies to generate multiple label refinements on the same event log are implemented and publicly available in the process mining toolkit ProM [38] as part of the *LabelRefinements*[1] package.

6. Related Work

We classify related work into three categories. The first category of related work concerns techniques from the process mining field, specifically focusing on techniques that, like our approach, focus on refining activity labels. The second category of related work, also originating from the process mining area, focuses on the interplay between ordering between process activities and external information, such as time. The third category of related work originates from the ambient intelligence and smart home environments field, focusing on work on mining temporal relations between human activities. We use these three categories to structure this section.

6.1. Label Splits and Refinements in Process Mining

The task of finding refinements of event labels in the event log is closely related to the task of mining process models with duplicate activities, in which the resulting process model can contain multiple transitions/nodes with the same label. From the point of view of the behavior allowed by a process model, it makes no difference whether a process model is discovered on an event log with refined labels, or whether a process model is discovered with duplicate activities such that each transition/node of the duplicate activity precisely covers one version of the refined label. However, a refined label may also provide additional

[1] https://svn.win.tue.nl/repos/prom/Packages/LabelRefinements/

insights as the new labels are explainable in terms of time. The first process discovery algorithm capable of discovering duplicate tasks was proposed by Herbst and Karagiannis in 2004 [39], after which many others have been proposed, including the Evolutionary Tree Miner [40], the α^*-algorithm [41], the $\alpha^\#$-algorithm [42], the EnhancedWFMiner [43]. An alternative approach has been proposed by Vázques-Barreiros [44] et al., who describe a local search based approach to repair a process model to include duplicate activities, starting from an event log and a process model without duplicate activities. Existing work on mining models with duplicate activities all base their duplicate activities on how well the event log fits the process model, and do not try to find semantic differences between the different versions of the activities in the form of attribute differences.

The work that is closest to our work is the work by Lu et al. [45], who describe an approach to pre-process an event log by refining event labels with the goal of discovering a process model with duplicate activities. The method proposed by Lu et al., however, does not base the relabelings on data attributes of those events and only uses the control flow context, leaving uncertainty whether two events relabeled differently are actually semantically different.

6.2. Data-Aware Process Mining

Another area of related work is data-aware process mining, where the aim is to discover rules with regard to data attributes of events that decide decision points in the process. De Leoni and van der Aalst [46] proposed a method that discovers data guards for decision points in the process based on alignments and decision tree learning. This approach relies on the discovery of a behaviorally well-fitting process model from the original event log. When only overgeneralizing process models (i.e., allowing for too much behavior) can be discovered from an event log, the correct decision points might not be present in the discovered process model at all, resulting in this approach not being able to discover the data dependencies that are in the event log. Our label refinements use data attributes prior to process discovery to enable the discovery of more behaviorally constrained process models by bringing parts of the event attribute space to the event label.

6.3. Temporal Relation Mining for Smart Home Environments

Galushka et al. [10] provide an overview of temporal data mining techniques and discuss their applicability to data from smart home environments. Many of the techniques described in the overview focus on real-valued time series data, instead of discrete sequences which we assume as input in this work. For discrete sequence data, Galushka et al. [10] propose the use of sequential rule mining techniques, which can discover rules of the form "*if event **a** occurs then event **b** occurs with time **T**"*.

Huynh et al. [9] proposed to use topic modeling to mine activity patterns from sequences of human events. Topic modeling originates from the field of natural language processing and addresses the challenge to find topics in textual documents and assign a distribution over these topics to each document. However, the discovered topics do not represent the human activities in terms of control-flow ordering constructs like sequential ordering, concurrent execution, choices, and loops.

Ogale et al. [11] proposed an approach to describe the temporal relation between human behavior activities from video data using context-free grammars, using the human poses extracted from the video as the alphabet. Like Petri nets, context-free grammars define a formal language over its alphabet. However, Petri nets have a graphical representation, which is lacking for grammars. Furthermore, as shown by Peterson [12], Petri net languages are a subclass of context-sensitive languages, and some Petri net languages are not context-free. This indicates that some relations over activities that can be expressed in Petri nets cannot be expressed in a context-free grammar.

One particularly related technique, called *TEmporal RElation Discovery of Daily Activities (TEREDA)*, was proposed by Nazerfard et al. [13]. TEREDA leverages temporal association rule mining techniques to mine ordering relations between activities as well as patterns in their timestamp and duration. The ordering relations between activities that are discovered by TEREDA are restricted to the form "*activity* **a** *follows activity* **b**", where our proposed approach of modeling the relations with Petri nets allow for modeling of more complex relations between larger number of activities, such as: "*the occurrences of activity* **b** *that are preceded by activity* **a** *are followed by both activity* **d** *and* **e***, but in arbitrary order"*. The patterns in the timestamps are obtained by fitting a Gaussian Mixture Model (GMM)

with the Expectation-Maximization (EM) algorithm, thereby ignoring problems caused by the circularity of the 24-hour clock introduced in this paper.

Jukkala and Cook [14] propose a method to mine temporal relations between activities from smart home environments logs where the temporal relation patterns are expressed in *Allen's interval algebra* [47]. Allen's interval algebra allows the expression of thirteen distinct types of temporal relations between two activities based on both the *start* and *end* timestamps of these activities. The approach of Jukkala and Cook [14] is limited to describing the relations between pairs of activities, and more complex relations between three or higher numbers of activities cannot be discovered. The aim of mining the patterns in Allen's interval algebra representation is to increase the accuracy of activity recognition systems, while our goal is knowledge discovery. Several papers from the process mining area have focused on mining temporal relations between activities from smart home event logs. Leotta et al. [5] postulate three main research challenges for the applicability of process mining technique for smart home data. One of those three challenges is to improve process mining techniques to address the less structured nature of human behavior as compared to business processes. Our technique addresses this challenge, as the time-based label refinements help in uncovering relations between activities with process mining techniques that could not be found without applying time-based label refinements.

DiMaggio et al. [6] and Sztyler et al. [2, 7] propose to mine Fuzzy Models [48] to describe the temporal relations between human activities. The Fuzzy Miner [48], a process discovery algorithm that mines a Fuzzy Model from an event log, is a process discovery algorithm that is designed specifically for weakly structured processes. However, Fuzzy Models, in contrast to Petri nets, do not define a formal language over the activities, and are therefore not precise on what activity orderings are allowed and which are not. While mining a Fuzzy Model description of human activities is less challenging compared to mining a process model with formal semantics, it is also limited in the insights that can be obtained from it.

Finally, insights in the human routines can be obtained through the discovery of *Local Process Models* [49], which bridges process mining and sequential pattern mining by finding patterns that include high-level process model constructs such as (exclusive) choices, loops, and concurrency. However, Local Process Models, as opposed to process discovery, only give insight into frequent subroutines of behavior and do not provide the global picture of the behavior throughout the day from start to end.

7. Conclusion & Future Work

We have proposed a framework based on techniques from the field of circular statistics to refine event labels automatically based on their timestamp attribute. We have shown on a real-life event log that this framework can be used to discover label refinements that allow for the discovery of more insightful and behaviorally more specific process models. Additionally, we explored four strategies to search combinations of label refinements. We found that the difference between an all-at-once strategy, which ignores that one label refinement can have an effect on the usefulness of other label refinements, and other more computationally expensive strategies is often limited. An interesting area of future work is to explore the use of other types of event data attributes to refine labels, e.g., power values of sensors. A next research step would be to explore label refinements based on a combination of data attributes combined. This introduces new challenges, such as the clustering on partially circular and partially Euclidean data spaces. Additionally, other time-based types of circles than the daily circle described in this paper, such as the week, month, or year circle, are worth investigating.

References

[1] W.M.P. van der Aalst, *Process mining: data science in action*, Springer Science & Business Media, 2016.

[2] T. Sztyler, J. Völker, J. Carmona, O. Meier and H. Stuckenschmidt, 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, 2015, pp. 22–23.

[3] N. Tax, N. Sidorova, R. Haakma and W.M.P. van der Aalst, Log-based evaluation of label splits for process models, *Procedia Computer Science* **96** (2016), 63–72.

[4] N. Tax, N. Sidorova, R. Haakma and W.M.P. van der Aalst, Event abstraction for process mining using supervised learning techniques, in: *Proceedings of the SAI Conference on Intelligent Systems*, Springer, 2016, pp. 161–170.

[5] F. Leotta, M. Mecella and J. Mendling, Applying process mining to smart spaces: Perspectives and research challenges, in: *Enterprise, Business-Process and Information Systems Modeling*, Springer, 2015, pp. 298–304.

[6] M. Dimaggio, F. Leotta, M. Mecella and D. Sora, Process-Based Habit Mining: Experiments and Techniques, in: *Proceedings of the IEEE International Conference on Ubiquitous Intelligence & Computing*, IEEE, 2016, pp. 145–152.

[7] T. Sztyler, J. Carmona, J. Völker and H. Stuckenschmidt, Self-tracking reloaded: applying process mining to personalized health care from labeled sensor data, in: *Transactions on Petri Nets and Other Models of Concurrency XI*, Springer, 2016, pp. 160–180.

[8] N. Tax, N. Sidorova, R. Haakma and W.M.P. van der Aalst, Mining Process Model Descriptions of Daily Life through Event Abstraction, in: *Intelligent Systems and Applications*, Springer, 2018, Chap. To Appear.

[9] T. Huynh, M. Fritz and B. Schiele, Discovery of activity patterns using topic models, in: *Proceedings of the 10th international conference on Ubiquitous computing*, ACM, 2008, pp. 10–19.

[10] M. Galushka, D. Patterson and N. Rooney, Temporal data mining for smart homes, in: *Designing Smart Homes*, Springer, 2006, pp. 85–108.

[11] A. Ogale, A. Karapurkar and Y. Aloimonos, View-invariant modeling and recognition of human actions using grammars, Springer, 2007, pp. 115–126.

[12] J.L. Peterson, *Petri net theory and the modeling of systems*, Prentice Hall, 1981.

[13] E. Nazerfard, P. Rashidi and D. Cook, Using association rule mining to discover temporal relations of daily activities, *Toward Useful Services for Elderly and People with Disabilities* (2011), 49–56.

[14] V.R. Jakkula and D.J. Cook, Using temporal relations in smart environment data for activity prediction, in: *Proceedings of the ICML Workshop on the Induction of Process Models*, 2007.

[15] W.M.P. van Der Aalst, A.H.M. Ter Hofstede, B. Kiepuszewski and A.P. Barros, Workflow patterns, *Distributed and parallel databases* **14**(1) (2003), 5–51.

[16] N. Tax, E. Alasgarov, N. Sidorova and R. Haakma, On Generation of Time-based Label Refinements, in: *Proceedings of the 25th International Workshop on Concurrency, Specification and Programming*, 2016.

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

[18] J.M.E.M. van der Werf, B.F. van Dongen, C.A.J. Hurkens and A. Serebrenik, Process discovery using integer linear programming, *Fundamenta Informaticae* **94**(3–4) (2009), 387–412.

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

[20] V. Liesaputra, S. Yongchareon and S. Chaisiri, Efficient process model discovery using maximal pattern mining, in: *Proceedings of the International Conference on Business Process Management*, Springer, 2015, pp. 441–456.

[21] A. Weijters and J. Ribeiro, Flexible heuristics miner (FHM), in: *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, IEEE, 2011, pp. 310–317.

[22] A. Augusto, R. Conforti, M. Dumas, M. La Rosa and G. Bruno, Automated discovery of structured process models: Discover structured vs. discover and structure, in: *Proceedings of the International Conference on Conceptual Modeling*, Springer, 2016, pp. 313–329.

[23] A.P. Dempster, N.M. Laird and D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society. Series B.* (1977), 1–38.

[24] T. van Kasteren, A. Noulas, G. Englebienne and B. Kröse, Accurate activity recognition in a home setting, in: *Proceedings of the 10th International Conference on Ubiquitous Computing*, ACM, 2008, pp. 1–9.

[25] K.V. Mardia and P.E. Jupp, *Directional statistics*, Vol. 494, John Wiley & Sons, 2009.

[26] J. Rao, Some tests based on arc-lengths for the circle, *Sankhyā: The Indian Journal of Statistics, Series B* (1976), 329–338.

[27] J.A. Hartigan and P.M. Hartigan, The dip test of unimodality, *The Annals of Statistics* (1985), 70–84.

[28] G. Schwarz, Estimating the dimension of a model, *The Annals of Statistics* **6**(2) (1978), 461–464.

[29] R.E. Kass and A.E. Raftery, Bayes factors, *Journal of the American Statistical Association* **90**(430) (1995), 773–795.

[30] T. Cohen and M. Welling, Harmonic exponential families on manifolds, in: *Proceedings of The 32nd International Conference on Machine Learning*, JMLR W&CP, 2015, pp. 1757–1765.

[31] A. Banerjee, I.S. Dhillon, J. Ghosh and S. Sra, Clustering on the unit hypersphere using von Mises-Fisher distributions, *Journal of Machine Learning Research* **6**(Sep) (2005), 1345–1382.

[32] K. Hornik and B. Grün, movMF: an R package for fitting mixtures of von Mises-Fisher distributions, *Journal of Statistical Software* **58**(10) (2014), 1–31.

[33] G.S. Watson, Goodness-of-fit tests on a circle. II, *Biometrika* **49**(1/2) (1962), 57–63.

[34] S.J.J. Leemans, D. Fahland and W.M.P. van der Aalst, Discovering block-structured process models from event logs containing infrequent behaviour, in: *Proceedings of the International Conference on Business Process Management*, Springer, 2013, pp. 66–78.

[35] A. Rozinat and W.M.P. van der Aalst, Conformance checking of processes based on monitoring real behavior, *Information Systems* **33**(1) (2008), 64–95.

[36] J. Munoz-Gama and J. Carmona, A fresh look at precision in process conformance, in: *Proceedings of the International Conference on Business Process Management*, Springer, 2010, pp. 211–226.

[37] E.M. Tapia, S.S. Intille and K. Larson, Activity recognition in the home using simple and ubiquitous sensors, in: *Proceedings of the International Conference on Pervasive Computing*, Springer, 2004, pp. 158–175.

[38] B.F. van Dongen, A.J.M.M. Weijters and W.M.P. van der Aalst, The ProM Framework: A New Era in Process Mining Tool Support, *Applications and Theory of Petri Nets* (2005), 444.

[39] J. Herbst and D. Karagiannis, Workflow mining with InWoLvE, *Computers in Industry* **53**(3) (2004), 245–264.

[40] J.C.A.M. Buijs, B.F. van Dongen and W.M.P. van der Aalst, On the role of fitness, precision, generalization and simplicity in process discovery, in: *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*, Springer, 2012, pp. 305–322.

[41] J. Li, D. Liu and B. Yang, Process mining: Extending α-algorithm to mine duplicate tasks in process logs, in: *Advances in Web and Network Technologies, and Information Management*, Springer, 2007, pp. 396–407.

[42] C.-Q. Gu, H.-Y. Chang and Y. Yi, Workflow mining: Extending the α-algorithm to mine duplicate tasks, in: *Proceedings of the International Conference on Machine Learning and Cybernetics*, Vol. 1, IEEE, 2008, pp. 361–368.

[43] F. Folino, G. Greco, A. Guzzo and L. Pontieri, Discovering expressive process models from noised log data, in: *Proceedings of the International Database Engineering & Applications Symposium*, ACM, 2009, pp. 162–172.

[44] B. Vázquez-Barreiros, M. Mucientes and M. Lama, Mining duplicate tasks from discovered processes, in: *Proceedings of the International Workshop on Algorithms & Theories for the Analysis of Event Data*, CEUR-ws.org, 2015, pp. 78–82.

[45] X. Lu, D. Fahland, F.J.H.M. van den Biggelaar and W.M.P. van der Aalst, Handling duplicated tasks in process discovery by refining event labels, in: *Proceedings of the International Conference on Business Process Management*, Springer, 2016, pp. 90–107.

[46] M. de Leoni and W.M.P. van der Aalst, Data-aware process mining: discovering decisions in processes using alignments, in: *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, ACM, 2013, pp. 1454–1461.

[47] J.F. Allen, Maintaining knowledge about temporal intervals, *Communications of the ACM* **26**(11) (1983), 832–843.

[48] C. Günther and W.M.P. van der Aalst, Fuzzy mining–adaptive process simplification based on multi-perspective metrics, in: *Proceedings of the International Conference on Business Process Management*, Springer, 2007, pp. 328–343.

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

[50] W. Reisig and G. Rozenberg, *Lectures on Petri nets I: basic models: advances in Petri nets*, Vol. 1491, Springer Science & Business Media, 1998.

[51] R Core Team, *R: a language and environment for statistical computing*, R Foundation for Statistical Computing, Vienna, Austria, 2013, ISBN 3-900051-07-0. http://www.R-project.org/.

[52] T. Benaglia, D. Chauveau, D. Hunter and D. Young, Mixtools: An R package for analyzing finite mixture models, *Journal of Statistical Software* **32**(6) (2009), 1–29.

[53] K. Liu, W.K. Cheung and J. Liu, Extracting Behavioral Motifs for Characterizing Human Daily Activities in Smart Environments, in: *Proceedings of the ACM SIGKDD Workshop on Health Informatics*, 2012, pp. 1–8.