On Generation of Time-based Label Refinements

Niek Tax[1,2], Emin Alasgarov[1,2], Natalia Sidorova[1], and Reinder Haakma[2]

[1] Eindhoven University of Technology, Department of Mathematics and Computer Science, P.O. Box 513, 5600MB Eindhoven, The Netherlands
{n.tax,n.sidorova}@tue.nl,{e.alasgarov}@student.tue.nl
[2] Philips Research, Prof. Holstlaan 4, 5665 AA Eindhoven, The Netherlands
{niek.tax,reinder.haakma}@philips.com

Abstract. Process mining is a research field focused on the analysis of event data with the aim of extracting insights in processes. Applying process mining techniques on data from smart home environments has the potential to provide valuable insights in (un)healthy habits and to contribute to ambient assisted living solutions. Finding the right event labels to enable 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 (overgeneralizing). Refinements of sensor level event labels suggested by domain experts have shown to enable discovery of more precise and insightful process models. However, there exist no automated approach to generate refinements of event labels in the context of process mining. In this paper we propose a framework for automated generation of label refinements based on the time attribute of events. We show on a case study with real life smart home event data that behaviorally more specific, and therefore more insightful, process models can be found by using automatically generated refined labels in process discovery.

Keywords: Label Refinements, Process Discovery, Unsupervised Learning

1 Introduction

Process mining is a fast growing discipline that combines knowledge and techniques from data mining, process modeling, and process model analysis [22]. Process mining techniques concern the analysis of events that are logged during process execution, where event records contain information on what was done, by whom, for whom, where, when, etc. Events are 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 have broadened in recent years from business process management to other application domains, one of them being analysis of

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	...	...
12	03/14/2015 03:41:46	Mountain Rd. 7	Motion sensor - Bedroom	1
13	03/14/2015 05:00:17	Mountain Rd. 7	Motion sensor - Bedroom	1
14	03/14/2015 08:52:32	Mountain Rd. 7	Motion sensor - Bedroom	1
15	03/14/2015 09:30:54	Mountain Rd. 7	Motion sensor - Living room	1
16	03/14/2015 09:35:25	Mountain Rd. 7	Power sensor - TV	160
17	03/14/2015 10:27:37	Mountain Rd. 7	Power sensor - TV	0
...	03/14/2015 ...	Mountain Rd. 7	...	...
...	...	...	...	...

events of human behavior with data originating from sensors in smart home environments [19, 21, 20]. 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 home, power sensors placed on appliances, open/close sensors placed on closets and cabinets, etc. Particularly challenging in applying process mining in this application domain is the extraction of meaningful event labels that allow for 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 [21]. Abstracting sensor-level events into events at the level of human activity (e.g. *eating*, *sleeping, etc.*) using techniques closely related to techniques used in the activity recognition field helps to discover more behaviorally more constrained and insightful process models [20], but applicability of this approach relies on the availability of a reliable diary of human behavior at the activity level, which is often just impossible to obtain.

In our earlier work [21] 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 [21] were based on domain knowledge, and not identified automatically from the data. In this paper, we aim at 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.

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 data set and show the effect of the refined event labels on process discovery. We continue by describing related work in Section 5 and conclude in Section 6.

2 Preliminaries

In this section we introduce the notions related to event logs and relabeling functions for traces and then define the notions of refinements and abstractions. We also introduce the Petri net process model notation.

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 $s : \{1, \ldots, n\} \to A$ 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. $\mathfrak{L}^p$ is the prefix closure of a language $\mathfrak{L}$ (with $\mathfrak{L} \subseteq \mathfrak{L}^p$).

An event is the most elementary element of an event log. Let $\mathcal{I}$ be a set of event identifiers, and $\mathcal{A}_1 \times \cdots \times \mathcal{A}_n$ be an attribute domain consisting of n attributes (e.g. timestamp, resource, activity name, cost, etc.). An event is a tuple $e = (i, a_1, \ldots, a_n)$, with $i \in \mathcal{I}$ 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 $(a_1 \ldots, a_n)$; e_i, and e_a respectively denote the identifier and label of event e. The timestamp attribute of an event is denoted by a_t. $\mathcal{E} = \mathcal{I} \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$ 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 *timestamp*, *sensor*, *address*, 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 any 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 $k, m \in \mathbb{N}$, $1 \leq k < m \leq |E|$, we have: $\sigma(k)_t \leq \sigma(m)_t$. 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 e.g. group together events with the same *address* and the same day-part of the *timestamp*, as indicated by the horizontal lines in Table 1. For each of these event sets, we can construct a trace; time stamps define the ordering of events within the trace. For events of a trace having the same time stamps, an arbitrary ordering can be chosen within a trace.

An *event partitioning function* is a function $ep : \mathcal{E} \to T_{id}$ 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 $L \subseteq \Sigma(\mathcal{E})$. $A_L \subseteq \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$ 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 *informative* and *repeatable* 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. Next to that, if the event partitioning function maps each event from Table 1 to its address and the day-part of the timestamp, these attributes (indicated in gray) become the trace attribute and can safely be removed from individual events.

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 differentiating between the mainstream behavior of a process (frequently occurring behavioral patterns) and exceptional behavior.

Let $\mathcal{E}, \mathcal{E}'$ be an event universe. A function $l : \mathcal{E} \to \mathcal{E}'$ is an *event relabeling function*. A relabeling function can be used to obtain more useful event labels than the full set of event attribute values. 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 $l_1 : \mathcal{E} \to \mathcal{E}_1$ and $l_2 : \mathcal{E} \to \mathcal{E}_2$ be event relabeling functions. Relabeling function l_1 is a *refinement* of relabeling function l_2, denoted by $l_1 \preceq l_2$, iff $\forall_{e_1,e_2 \in \mathcal{E}} : l_1(e_1) = l_1(e_2) \implies l_2(e_1) = l_2(e_2)$; 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. A frequently used process modeling notation in the process mining field is the Petri net [16]. Petri nets are directed bipartite 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 execution 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 $\ell : T \to A_M \cup \{\tau\}$ 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 $\bullet n = \{n | (n', n) \in F\}$ and $n\bullet = \{n | (n, n') \in F\}$. An example of a Petri net can be seen in Figure 1, where circles represent places and squares represent transitions. Gray transitions with smaller width represent 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

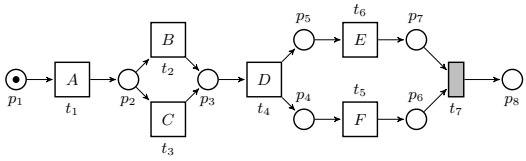

Fig. 1. An example Petri net.

3-tuple (N, M_i, M_f) with N a labeled Petri net, M_i an initial marking, and M_f a set of final markings. Many process modeling notations, including accepting Petri nets, have formal executional semantics and a model defines a *language of accepting traces* $\mathfrak{L}$. For the Petri net in Figure 1, the language of accepting traces is $\{\langle A, B, D, E, F\rangle, \langle A, B, D, F, E\rangle, \langle A, C, D, E, F\rangle, \langle A, C, D, F, E\rangle\}$.

3 A Framework for Time-based Label Refinements

To generate potential label refinements for every label based on time we take a clustering based approach by identifying dense areas in time space for 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 hourfloat representation in 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) 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 example of a 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, as shown in Figure 2. The GMM fitted to the timestamps of the sensor events consists of two components, one with the mean at 9.05 and one with a mean at 20. The histogram representation of the same data shows that some events happened just after midnight, which is actually closer on the clock to 20 than to 9.05. The GMM however is unaware of the circularity of the clock, which results in the mixture model that seems inappropriate when visually comparing with the histogram. The field of circular statistics (also referred to as directional statistics), concerns analysis of such circular data spaces (cf. [14]).

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:

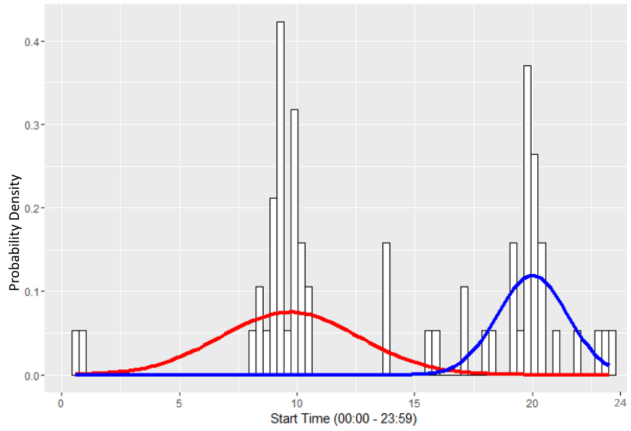

Fig. 2. The histogram representation and a Gaussian Mixture Model fitted to timestamps values of the plates cupboard sensor in the van Kasteren data set [23].

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 how many clusters the events of a sensor type contain.

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.

Data-model post-fitting stage In this stage the quality of the label refinements is assessed from both a cluster quality perspective and a process model (event ordering statistics) perspective.

3.1 Data-model pre-fitting stage

We now describe a test for uniformity, a test for unimodality, and a method to select the number of clusters in the data.

Uniformity Check - Rao's Spacing Test Rao's spacing test [15] 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 $\frac{360}{n}$ degrees. 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 $\lambda = \frac{360^\circ}{n}$, $T_i = f_{i+1} - f_i$ for $1 \leq i \leq n-1$ and $T_n = (360^\circ - f_n) + f_1$.

Unimodality Check - Hartigan's Dip Test Hartigan's dip tests [7] 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 the empirical distribution function and the unimodal distribution function that minimizes that maximum difference.

Number of Component Selection - Bayesian Information Criterion The Bayesian Information Criterion (BIC) [17] 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 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 1 component, and iteratively increase from k to $k+1$ components as long as the decrease in BIC is larger than 10, which is the threshold for decisive evidence of high BIC [10].

3.2 Data-model fitting stage

We cluster events generated by one sensor using a mixture model consisting of components of the von Mises distribution, which is a circular version of the normal distribution. This technique is based on the approach of Banerjee et al. [1], who introduce 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 $0 \leq \theta \leq 2\pi,\ 0 \leq \mu \leq 2\pi,\ \kappa \geq 0$. 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 [9] provided in R.

3.3 Data-model post-fitting stage

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 [25], 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 $U^2 = n \int_0^{2\pi} \left[F_n(\theta) - F(\theta) - \int_0^{2\pi} \left\{ F_n(\phi) - F(\phi) \right\} dF(\phi) \right]^2 dF(\theta)$.

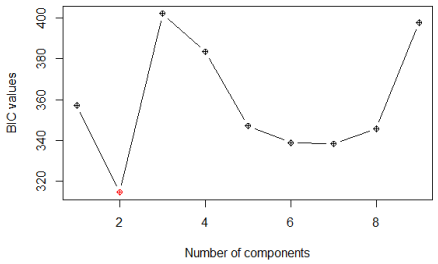

Fig. 3. BIC values for different numbers of components in the mixture model.

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

Furthermore we assess the quality of refining the event label into a new label for each cluster from a process perspective using the label refinement evaluation method described in [21]. This method tests whether the log statistics that are used in many process discovery algorithms become significantly more deterministic by applying the label refinement.

4 Case Study

We show the results of our time-based label refinements approach on the real life smart home data set described in van Kasteren et al. [23]. The van Kasteren data set consists of 1285 events divided over fourteen different sensors. We segment in days from midnight to midnight to define cases. Figure 4a shows the process model discovered on this event log with the Inductive Miner infrequent [11] with 20% filtering, which discovers a process model that describes the most frequent 80% of behavior in the log. Note that this process model overgeneralises allowing too much behaviour. 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 our proof of concept by applying the framework 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 3 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 $0 = 2\pi$ 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

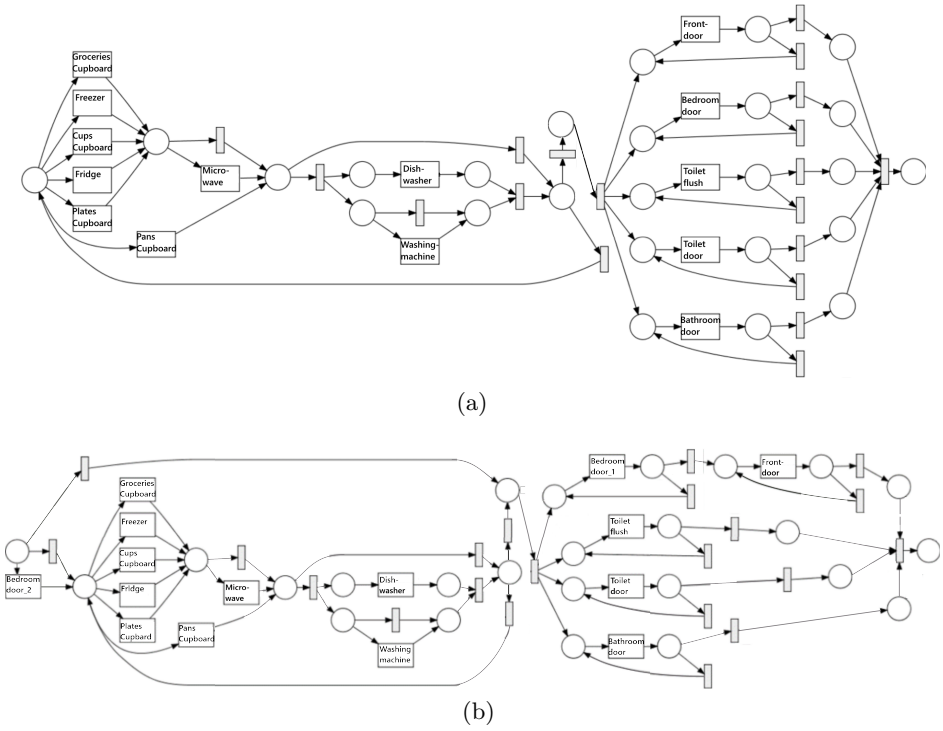

Fig. 4. Process models discovered on the van Kasteren data with sensor-level labels (a) and refined labels (b) with the Inductive Miner infrequent (20% filtering) [11].

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 likely to be generated by the two von Mises distributions found. The label refinement evaluation method [21] 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 4b 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 in general, but the label refinement does help restricting 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 4a *bedroom door* and *frontdoor* events can occur an arbitrary number of

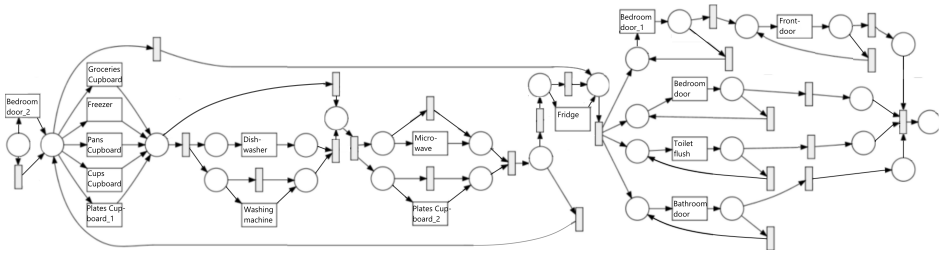

Fig. 5. Inductive Miner infrequent (20% filtering) [11] result after a second label refinement.

times in any order. Figure 4a furthermore does not allow for the *bedroom door* to occur before the whole block of kitchen-located events at the beginning of the net.

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 in directly before or after the microwave.

5 Related Work

Refining 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 versions of the refined label. The first process discovery algorithm capable of discovering duplicate tasks was proposed by Herbst and Karagiannis in 2004 [8], after which many others have been proposed, including the Genetic Miner [4], the Evolutionary Tree Miner [2], the α^*-algorithm [12], the $\alpha^\#$-algorithm [6], the EnhancedWFMiner [5], and a simulated annealing based algorithm [18]. An alternative approach has been proposed by Vázques-Barreiros [24] 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 any semantic difference between the multiple versions of the activities in the form of data attribute differences.

The work that is closest to our work is the work by Lu et al. [13], 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 but instead bases them solely on the control flow context, leaving uncertainty whether two events relabeled differently are actually semantically different.

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 [3] 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 discover more behaviorally constrained process models by bringing parts of the event attribute space to the event label.

6 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 through a proof of concept on a real life event log that this framework can be used to discover label refinements that allow for discovery of more insightful and behaviorally more specific process models. 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 multiple data attributes combined. This would bring challenge of clustering on partially circular and partially euclidean data spaces.

References

1. Banerjee, A., Dhillon, I. S., Ghosh, J., and Sra, S. Clustering on the unit hypersphere using von mises-fisher distributions. *Journal of Machine Learning Research 6*, Sep (2005), 1345–1382.
2. Buijs, J. C. A. M., Van Dongen, B. F., and van der Aalst, W. M. P. On the role of fitness, precision, generalization and simplicity in process discovery. In *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"* (2012), Springer, pp. 305–322.
3. de Leoni, M., and van der Aalst, W. M. P. Data-aware process mining: discovering decisions in processes using alignments. In *Proceedings of the 28th annual ACM symposium on applied computing* (2013), ACM, pp. 1454–1461.
4. de Medeiros, A. K. A., Weijters, A. J. M. M., and van der Aalst, W. M. P. Genetic process mining: an experimental evaluation. *Data Mining and Knowledge Discovery 14*, 2 (2007), 245–304.
5. Folino, F., Greco, G., Guzzo, A., and Pontieri, L. Discovering expressive process models from noised log data. In *Proceedings of the 2009 international database engineering & applications symposium* (2009), ACM, pp. 162–172.
6. Gu, C.-Q., Chang, H.-Y., and Yi, Y. Workflow mining: Extending the α-algorithm to mine duplicate tasks. In *2008 International Conference on Machine Learning and Cybernetics* (2008), vol. 1, IEEE, pp. 361–368.

7. Hartigan, J. A., and Hartigan, P. M. The dip test of unimodality. *The Annals of Statistics* (1985), 70–84.
8. Herbst, J., and Karagiannis, D. Workflow mining with inwolve. *Computers in Industry 53*, 3 (2004), 245–264.
9. Hornik, K., and Grün, B. movmf: An r package for fitting mixtures of von mises-fisher distributions. *Journal of Statistical Software 58*, 10 (2014), 1–31.
10. Kass, R. E., and Raftery, A. E. Bayes factors. *Journal of the American Statistical Association 90*, 430 (1995), 773–795.
11. Leemans, S. J. J., Fahland, D., and van der Aalst, W. M. P. Discovering block-structured process models from event logs containing infrequent behaviour. In *International Conference on Business Process Management* (2013), Springer, pp. 66–78.
12. Li, J., Liu, D., and Yang, B. 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.
13. Lu, X., Fahland, D., van den Biggelaar, F. J. H. M., and van der Aalst, W. M. P. Handling duplicated tasks in process discovery by refining event labels. In *International Conference on Business Process Management* (2016), Springer, p. To appear.
14. Mardia, K. V., and Jupp, P. E. *Directional statistics*, vol. 494. John Wiley & Sons, 2009.
15. Rao, J. Some tests based on arc-lengths for the circle. *Sankhyā: The Indian Journal of Statistics, Series B* (1976), 329–338.
16. Reisig, W., and Rozenberg, G. *Lectures on Petri nets I: basic models: advances in Petri nets*, vol. 1491. Springer Science & Business Media, 1998.
17. Schwarz, G. Estimating the dimension of a model. *The Annals of Statistics 6*, 2 (1978), 461–464.
18. Song, W., Liu, S., and Liu, Q. Business process mining based on simulated annealing. In *Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for* (2008), IEEE, pp. 725–730.
19. Sztyler, T., Völker, J., Carmona, J., Meier, O., and Stuckenschmidt, H. 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 (ATAED)* (2015), pp. 22–23.
20. Tax, N., Sidorova, N., Haakma, R., and van der Aalst, W. M. P. Event abstraction for process mining using supervised learning techniques. In *Proceedings of the SAI Conference on Intelligent Systems (IntelliSys)* (2016), IEEE, pp. 161–170.
21. Tax, N., Sidorova, N., Haakma, R., and van der Aalst, W. M. P. Log-based evaluation of label splits for process models. *Procedia Computer Science* (2016), To appear.
22. van der Aalst, W. M. P. *Process mining: data science in action*. Springer Science & Business Media, 2016.
23. van Kasteren, T., Noulas, A., Englebienne, G., and Kröse, B. Accurate activity recognition in a home setting. In *Proceedings of the 10th International Conference on Ubiquitous Computing* (2008), ACM, pp. 1–9.
24. Vázquez-Barreiros, B., Mucientes, M., and Lama, M. Mining duplicate tasks from discovered processes. In *Proceedings of the International Workshop on Algorithms & Theories for the Analysis of Event Data (ATAED)* (2015), pp. 78–82.
25. Watson, G. S. Goodness-of-fit tests on a circle. ii. *Biometrika 49*, 1/2 (1962), 57–63.