Handling Duplicated Tasks in Process Discovery by Refining Event Labels

Xixi Lu[1(✉)], Dirk Fahland[1], Frank J.H.M. van den Biggelaar[2], and Wil M.P. van der Aalst[1]

[1] Eindhoven University of Technology, Eindhoven, The Netherlands
{x.lu,d.fahland,w.m.p.v.d.aalst}@tue.nl
[2] Maastricht University Medical Center, Maastricht, The Netherlands
f.vanden.biggelaar@mumc.nl

Abstract. Processes may require to execute the same *activity* in different stages of the process. A human modeler can express this by creating two different *task* nodes labeled with the same activity name (thus *duplicating* the task). However, as events in an event log often are labeled with the activity name, discovery algorithms that derive tasks based on labels only cannot discover models with duplicate labels rendering the results imprecise. For example, for a log where "payment" events occur at the beginning and the end of a process, a modeler would create two different "payment" tasks, whereas a discovery algorithm introduces a loop around a single "payment" task. In this paper, we present a general approach for refining labels of events based on their context in the event log as a preprocessing step. The refined log can be input for any discovery algorithm. The approach is implemented in ProM and was evaluated in a controlled setting. We were able to improve the quality of up to 42 % of the models compared to using a log with imprecise labeling using default parameters and up to 87 % using adaptive parameters. Moreover, using our refinement approach significantly increased the similarity of the discovered model to the original process with duplicate labels allowing for better rediscoverability. We also report on a case study conducted for a Dutch hospital.

1 Introduction

Real-life processes may require that the same *activity* occurs at different stages or branches of the process [1–4]. A human modeler would use different *nodes* in a model (e.g., different transitions in a Petri net) labeled with the same activity to express different occurrences of an activity in the process. We call each node labeled with an activity a *task*. Thus, there could be many tasks referring to the same activity, which are known as *duplicated tasks*. In a log, events are usually labeled with activity names instead of tasks. As a result, two different events with the same activity label may originate from the same task or from different tasks, i.e., the labeling in the event log is *imprecise*.

Process discovery aims at creating an accurate representation of the real process from an event log helping users to understand the executed process [3,5].

M. La Rosa et al. (Eds.): BPM 2016, LNCS 9850, pp. 90–107, 2016.
DOI: 10.1007/978-3-319-45348-4_6

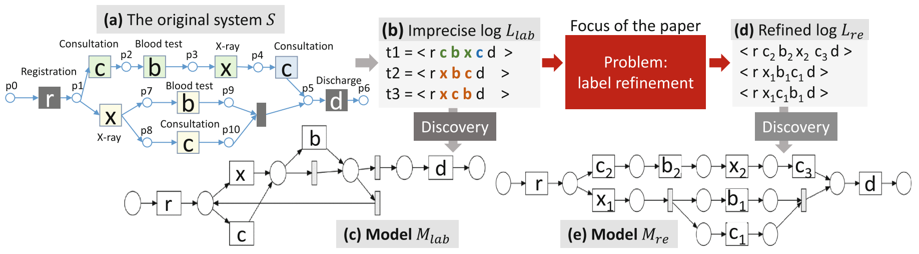

Fig. 1. The imprecise label problem settings and the running example.

However, most existing discovery algorithms assume the labels of events to be precise and consider for each label as one task represented by a single task node in the model. In case of event logs with imprecise labels, these discovery algorithms tend to return over-generalized models that allow much more behavior than in the event log [2,3]. Such models may be misleading or even incorrect, obstructing users to use the models for understanding the real processes or performing accurate process analysis. A better solution would be to discover models where two tasks carry the same label, i.e., *duplicate* tasks [1,2].

We exemplify the problem using an example shown in Fig. 1. The original system (a) has five activities "*r*", "*d*", "*c*", "*b*" and "*x*" and ten tasks; activities "*c*", "*b*" and "*x*" occur at multiple different tasks, which result in an imprecise log (b), in which the events only refer to activities "*c*", "*b*" and "*x*" rather than the different tasks in the system. Using a standard discovery algorithm, we discover for the imprecise log (b) an imprecise model (c) that states "*b*" could be skipped and has a loop that allows "*c*" and "*x*" to be executed an arbitrary number of times, even though every trace in the log has an event labeled "*b*" and only one event labeled "*x*". Overall, model (c) is imprecise as it contains many behaviors neither seen in the log (b), nor in the original system (a). Refining the labels of events could yield the refined log (d), from which a refined model (e) can be discovered that corresponds to the original model (a) while using the same discovery algorithm. However, the trivial refinement where each event gets its own unique label is not desired as it would lead to models that overfit the event log. Thus, our goal is to refine an imprecise log in such a way that a discovery algorithm finds a better model which is more precise and closer to the original model.

In this paper, we investigate the problem of imprecise labels of events for process discovery and propose an approach to resolve the problem through log preprocessing. In particular, we introduce an approach for refining labels and relabeling events in the log such that any existing or future process discovery algorithm can infer duplicated tasks from the refined labels. As the optimal refined log or model may be unknown, our approach aims at adding more alternative representations of a process into the solution space of process discovery algorithms to help users find better models systematically.

Our approach has three steps: (1) identify one or multiple candidates for imprecise event labels; then refine imprecise labels (2) across traces and (3) within traces. Here, we leverage previous work on trace matching technique which groups events based on similarities in their context [6]; dissimilar groups of events are labeled differently.

The approach is implemented in ProM[1] and has been evaluated in a controlled setting and in a real life case study. In the controlled experiment, we investigated how well our approach can detect and refine labels in imprecise event logs generated from a large set of synthetic process models with duplicated tasks. We analyzed model quality with respect to the event log and the similarity to the original model. For 87 % of the processes having duplicated tasks outside of loops, our approach automatically refined imprecise logs so that a discovery algorithm returned a more precise model. For processes having duplicated tasks in a loop, label refinement improved precision for 61 % of the imprecise logs.

In the remainder, we first discuss related work in Sect. 2. In Sect. 3, we recall the concepts used for defining the problem, the measures used in the evaluation, and the methods used in the approach. In Sect. 4, we formalize problems and aims. Section 5 explains the proposed approach. The evaluation results are presented in Sect. 6, and Sect. 7 concludes the paper.

2 Related Work

Process Model Elements Labeling or Relabeling. Many studies have investigated the problem of labeling or relabeling process elements (e.g., activities, flow relations) in process models [7,8]. These works assume that a collection of structurally correct process models is available and use additional domain knowledge or other semantically correct labels to then suggest or revise the incorrect labels of elements in these models. Here, we assume no models to be available and operate solely on event logs, in order to discover structurally correct models. One then may apply [7,8] on the discovered models to revise and improve their labels.

Process Discovery and Duplicated Tasks. Process discovery algorithms aim at discovering "good" models from an event log to help users to understand real-life processes. Most existing discovery algorithms map each unique event label to one task, making it impossible to discover processes with two tasks with the same label. Some discovery algorithms can refine labels during model construction to some extent [1,2,4,5,9]. However, these internal mechanism to handle duplicated tasks can not be used in other discovery algorithms. Moreover, these algorithms have other limitations such as they do not guarantee sound models or fitting models. To be able to benefit from current and future progress in process discovery techniques [10,11], we propose to refine labels in the event log itself, which then can be used by any process discovery algorithm.

[1] http://www.promtools.org/.

Trace Clustering and Clone Detection. As duplicated tasks may also manifest themselves as multiple variants of executing a set of activities within the same process, trace clustering was proposed as a way to distinguish these variants [12,13]. However, clustering techniques always consider entire traces and thus also unnecessary duplicate tasks which are the same in all variants. In [14], the authors proposed a top-down approach that clusters the traces, discovers models for each cluster separately and uses clone detection to find tasks that are the same in all variants, preventing unnecessary duplicating tasks [15]. However, trace clustering techniques are unable to distinguish two events having the same label within a trace or a variant [12]. In this paper, we aim at tackling both problems.

Data Quality and Noise/Deviation Filtering. Imprecise labels could also be seen as data quality problem, i.e., events having incorrect labels. To the best of our knowledge, no existing work investigated this problem from this point of view. Other existing work on log preprocessing such as noise/deviation filtering would change input logs, both structurally and behaviorally, e.g., by removing events [6]. Such changes would also affect fitness of the discovered model with respect to the original log as a process discovery algorithm can only guarantee fitness for the filtered log. In this paper, we propose to not change the event log but only the labeling of events, which help us to preserve fitness if the discovery algorithm has such a guarantee.

Model Quality of Discovered Models. Dozens criteria and measures have been proposed for assessing the quality of discovered models, which may be discussed in three categories. Measures that evaluate the quality of the model using the input log often consider fitness, precision, and generalization [3,5]; we use the fitness defined in [16] and precision in [17]. In the context of controlled experiments, the quality of a discovered model can be evaluated against the original system in terms of how much of the behavior of the system can be reproduced by the discovered model and how precise the model describes the system [18]. When evaluating the quality of model irrespective of the log, then soundness and simplicity are often considered [5,19]. In the next section, we further discuss the measures used in this paper.

3 Preliminaries

In this section, we present (1) the input for our approach, (2) the quality measures used, and (3) the key concepts of a technique for finding events with a similar context.

Event, Label, and Event Log. Let $\mathcal{E}$ be the universe of unique events, i.e., the set of all possible event identifiers. A trace $\sigma \in \mathcal{E}^*$ is a finite sequence of events. An event log $C = \{\sigma_1, \sigma_2, \cdots, \sigma_n\} \subseteq \mathcal{E}^*$ is a set of traces. Here we assume no event appears twice in the same trace nor in different traces. We use E_C for the set of events in log C. Let A be a set of activities and C a log. A *labeling* function

$l : E_C \rightarrow A$ is surjective and assigns to each event $e \in E_C$ a label $l(e) = a \in A$. We call $L = (C, l)$ a *labeled event log* over activities A.

Process Discovery and Model Quality. Let $L = (C, l)$ be a labeled log over A. A discovery algorithm D returns a model M (i.e. $D(L) = M$) such that the activities $\mathcal{A}(M)$ occurring in model M are A, i.e., $\mathcal{A}(M) = A$. The quality of the discovered model $D(L) = M$ may be evaluated in two ways. First, with respect to the input log L, the $log\_fitness(L, M)$ and $log\_precision(L, M)$ of the model can be computed, for which we use the measures defined in [16,17], respectively. Both return a value between 0 and 1: if $log\_fitness(L, M) = 1$, every trace in the log can be replayed by the model perfectly. When $log\_precision(L, M)$ is close to 1, most (alternative) behavior allowed by the model is also observed in the log.

In addition, we compare the discovered $M = D(L)$ to the original system in terms of *system recall* and *system precision* to evaluate the generalization and discoverability of our approach. Let S be the system that generated L. The system recall $sys\_recall(S, M, L)$ and system precision $sys\_precision(S, M, L)$ of the discovered model are computed according to [18]. For example, in Fig. 1(b), after executing events "r" and "x" in trace t_2, tasks "b" and "c" are enabled in the original system; in model (c), "b" and "c" but also "x" are enabled, which has 100 % recalled all enabled activities in the system but is less precise (an additional "x" not enabled in the system); a trace model would only allow "b" (the next event in the trace), which is precise but has bad recall. Note that the recall with respect to system thus also captures the aforementioned generalization quality [5]. Furthermore, note that the system precision is different from the log precision, as the system could be imprecise with respect to the log (when the log is incomplete), but system is always precise with respect to to itself.

Similar Events, Mapping, and Cost Function. We build on existing concepts [6] to identify events that carry the same label but occur in different contexts. Let σ, σ' be two traces. A *mapping* $\lambda_{(\sigma,\sigma')} \subseteq E_\sigma \times E_{\sigma'}$ between σ and σ' is a binary, injective relation; $(e, e') \in \lambda_{(\sigma,\sigma')}$ is a *matched pair*. We write $\overline{\lambda}_{(\sigma,\sigma')}$ for the set of events having not match in $\lambda_{(\sigma,\sigma')}$, i.e. $\overline{\lambda}_{(\sigma,\sigma')} = \{e \in E \mid \neg\exists e' \in E' : (e, e') \in \lambda_{(\sigma,\sigma')}\} \cup \{e' \in E' \mid \neg\exists e \in E : (e, e') \in \lambda_{(\sigma,\sigma')}\}$. In this paper, we assume for all $(e, e') \in \lambda_{(\sigma,\sigma')}$, $l(e) = l(e')$.

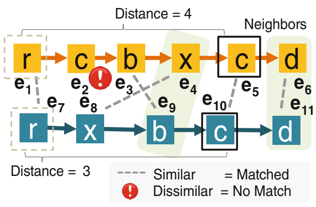

Fig. 2. An example of a mapping between two traces.

Given any two traces σ and σ', there are many possible mappings between them. An optimal mapping that maximizes the pairs of mapped events with large similarity in their context can be selected using a cost function with three weighted components: (1) the differences in the (direct or indirect) neighbors of the matched pairs (using $cost_{Matched}$), (2) the differences in the distances between a matched pair (e, e') and other matched pairs (using $cost_{Struc}$), and (3) the non-matched events $e \in \overline{\lambda}$ (using $cost_{NoMatch}$). Formally, $cost(\sigma, \sigma', \lambda) = w_M * \sum_{(e,e')\in\lambda} cost_{Matched}(e, e', \lambda) + w_S * \sum_{(e,e')\in\lambda} cost_{Struc}(e, e', \lambda) + w_N *$

$\sum_{e\in\overline{\lambda}} cost_{NoMatch}(e)$, in which w_M, w_S, w_N are the weights for the components. Figure 2 shows an example of a mapping between traces t_1 and t_2 of log L_{lab} of Fig. 1 (see [6] for a detailed explanation). As the traces σ and σ' are finite, one may simply enumerate all possible mappings between two traces, compute the cost of each mapping, and select the optimal ones. In [6], a greedy algorithm is proposed to find a locally optimal mapping in polynomial time. The results of this paper have been obtained with the greedy variant.

4 Problem Definition and Analysis

In this section, we first formally define our research problem and then discuss the related complications and our design decisions. In essence, given an imprecisely labeled log $L = (C, l_A)$ over the set of activities A, we would like to return a more *refined labeling function* l_B for the events E_C in order to help a discovery algorithm find better models.

Definition 1 (Refined Labeling Function). *For a labeled log L_A over the set of labels A, the log L_B over an arbitrary set of labels B is a* refined log *iff (1) they have the same traces, i.e., $L_A = (C, l_A), L_B = (C, l_B)$, and (2) for each two events $e, e' \in E_c$, e and e' can only have the same label according to l_B, if they also have the same label by l_A, i.e., $(l_B(e) = l_B(e')) \Rightarrow (l_A(e) = l_A(e'))$. We call l_B a* refined labeling function *for L_A.*

Note that the model M_B discovered from L_B has a different set of activity labels (i.e., $\mathcal{A}(M_B) = B$) than the model M_A discovered from L_A (i.e., $\mathcal{A}(M_A) = A$). However, for comparing M_A and M_B w.r.t. various measures, both models should have the same set of activities. To allow for this comparison, we introduce some notions that allow replacing the refined labels B of M_B with the original labels in A. Each refined log $L_B = (C, l_B)$ of $L_A = (C, l_A)$ induces the *label abstraction* function $\beta : B \rightarrow A$ with $\beta = \bigcup_{e\in E}\{l_B(e) \mapsto l_A(e)\}$. The inverse $\beta^{-1}(a) = \{b \in B | \beta(b) = a\}$ gives the set of refined labels for original label $a \in A$. Note that $\beta(l_B(e)) = l_A(e)$ for all events $e \in E$. For example, in Fig. 1, the refined log L_{re} of L_{lab} induces the abstraction $\beta = \{r \rightarrow r, c_1 \rightarrow c, c_2 \rightarrow c, c_3 \rightarrow c, b_1 \rightarrow b, b_2 \rightarrow b, x_1 \rightarrow x, x_2 \rightarrow x, d \rightarrow d\}$, label c is refined into the set $\beta^{-1}(c) = \{c_1, c_2, c_3\}$.

Using β, we can abstract model M_B by replacing each label b in M_B with $\beta(b)$. Let $\beta(M_B)$ denote the resulting model. Lemma 1 then follows immediately from the definitions.

Lemma 1 (M_B and $\beta(M_B)$ have the Same Behaviors). *Let L_A be an event log and L_B be a refined log of L_A. Let M_B be a model discovered from L_B such that each trace of L_B is a trace of M_B. Let β be the label abstraction induced by L_B. Then each trace of L_A is a trace of $\beta(M_B)$.*

Through β we can now compare models M_A and $\beta(M_B)$ respectively discovered from both original log L_A and refined log L_B and formally define our problem.

Problem Definition. Let $L_{lab} = (C, l)$ be an (imprecisely) labeled event log over the set of activities A. Let S denote the system model that generated L_{lab} with $\mathcal{A}(S) = A$. Given discovery algorithm D, let $M_{lab} = D(L_{lab})$ be the model discovered on the labeled log. We would like to find a *refined labeling function* l' of l that with induced label abstraction β such that for the refined log $L_{re} = (C, l')$ and the discovered, abstracted model $M_{re} = \beta(D(L_{re}))$ over A, the following properties hold:

(1) Fitness and precision of M_{re} improves over M_{lab} w.r.t. the given labeled log:
  - $log\_precision(M_{re}, L_{lab}) \geq log\_precision(M_{lab}, L_{lab})$ and
  - $log\_fitness(M_{re}, L_{lab}) \geq log\_fitness(M_{lab}, L_{lab})$

(2) Recall and precision of behavior of M_{re} should be higher than M_{lab} w.r.t. S, i.e.,
  - $sys\_precision(S, M_{re}, L_{lab}) \geq sys\_precision(S, M_{lab}, L_{lab})$ and
  - $sys\_recall(S, M_{re}, L_{lab}) \geq sys\_recall(S, M_{lab}, L_{lab})$

When the system S is unknown, we consider our third aim as providing different refined labeling functions that satisfy the first requirement which allows users to explore different representations of the input log.

Related Issues and Design Decisions. We discuss three complications related to the research problem to motivate our design decisions and assumption. First, there is a large combinatorial number of possible solutions, since in principle any label can be refined into an arbitrary number of refined labels. In addition, we have no criteria nor metrics that define when a refinement is optimal for the algorithm D. This depends on the discovery algorithm used. Furthermore, when the system is unknown, the same process may have different equally good representations depending on the stakeholder, the context, and decisions made in the formalization of the model. Since one can not deduce the optimal log nor the optimal model, we have to base the decisions for refining event labels on the behavioral structure of the event log and some basic principles and heuristics we discuss later.

The second complication is posed by the discovery algorithms and measures used for evaluation. Ideally, a more precise log would result in a more precise model, independent of the discovery algorithm and the measures we applied. However, this is not the case. A discovery algorithm may return a less precise model while the log is more refined (for example to avoid overfitting). Therefore, we decided to propose a backup plan. If $log\_precision(M_{re}, L_{lab}) < log\_precision(M_{lab}, L_{lab})$, we simply return the log with its imprecise labeling. This guarantees that at least using the refined log would not lead to discovering a model worse than using the imprecise log.

Finally, in this paper, we assume the discovered model is sound and fitting for the following reasons. Most state-of-art measures assume a fitting log when evaluating the quality of a model. We observed in our own experiments that the measures become rather unreliable and difficult to compare or to understand the improvements when the models are not sound and fitting. Moreover, as fitness is defined in terms of the number of events that can be replayed by a model and we are not adding or removing any events (which would have direct influence on the fitness), changes in fitness are merely a quality of the discovery algorithms used. For example, if the algorithm guarantees to return a fitting model, relabeling events would not change this property.

5 Approach

We decompose the label refinement problem into three subproblems. First, we identify one or multiple labels as candidates for imprecise labels. Then, we consider a group of traces that have *similar behavior* to be a *variant* of the process and refine the imprecise label candidates (horizontally) into different variants and (vertically) within a variant. Figure 3 shows an overview of the three subproblems using an example.

5.1 Detecting Imprecise Labels

The first step is to identify one or multiple candidates for imprecise labels. This step helps to limit the search scope to those events that have an imprecise label and to avoid splitting non-duplicated tasks. Furthermore, it helps to consolidate the context information of events with imprecise labels. One may also consider all labels, however, this may unnecessarily complicate the label refinement process.

Formally, we define the problem as follows. Let $L = (C, l_A)$ be a labeled event log and A the set of labels used. We would like to identify a subset of labels $A' \subseteq A$ and consider them as candidates for imprecise labels. In other words, the labels in $A \backslash A'$ are precise labels, and there is no need to refine them, i.e., for $e \in E_C$ and $l_A(e) = a \in A \backslash A'$, any refined labeling function l_B of l_A with its β implies $l_B(e) = a$, and $\beta^{-1}(a) = \{a\}$.

There are many different ways to detect imprecise labels. We discuss two methods (used in the evaluation) and consider other possibilities as future work. The optimal case is to have an oracle that returns the truly imprecise labels as candidates. For example, domain experts indicate a particular label to be imprecise. In the remainder, we refer to this as Oracle Detection (OD).

Besides having an oracle, we propose an automated method that uses properties of Inductive Miner (IM) [11]. IM systematically parses an event log and

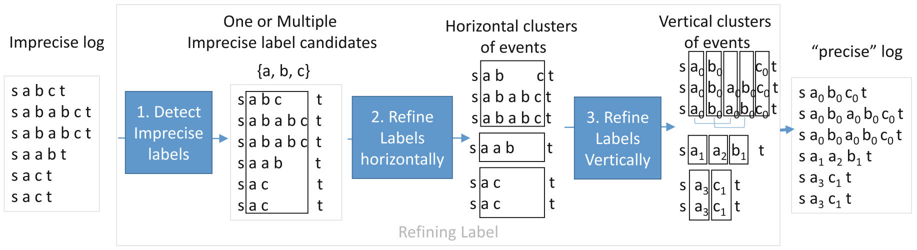

Fig. 3. The proposed approach for refining imprecise label as log preprocessing.

finds a locally optimal "subprocess" recursively. If IM fails to find an accurate subprocess, it returns a generic subprocess that can replay any trace over the events in the corresponding sublog (i.e., a local "flower loop"). In this paper, we consider this type of subprocess to be imprecise. We choose to select the smallest imprecise subprocess (i.e., local "flower loop") and return the activity labels in the subprocess as imprecise label candidates. For instance, applying IM on the running example, IM returns a process model of Fig. 1(c) containing a flower loop with activity labels c, b and x, and this set $\{c, b, x\}$ is returned as candidates for imprecise labels. We use the IM Detection (IMD) to refer to this method. In principle, any subprocess or multiple subprocesses can be selected.

5.2 Intermediate Step - Matching Events

After finding imprecise label candidates, we propose an intermediate step before refining these labels. The objective of this intermediate step is to identify similarities between events across traces. Similar events should carry the same refined label whereas dissimilar events should carry a different label.

In essence, the procedure for computing the similarity of events of different traces uses the existing trace matching technique of Sect. 3 and goes as follows. Given a labeled log $L = (C, l)$, for each two traces $\sigma, \sigma' \in C$, we find an optimal mapping $\lambda_{\sigma,\sigma'} \in E_\sigma \times E_{\sigma'}$ between their events for a given cost function. This way we get the distance between any two traces σ and σ' as $cost(\sigma, \sigma', \lambda_{\sigma,\sigma'})$. This distance can be normalized w.r.t. the highest cost $maxCost = \max_{\sigma,\sigma' \in C}\ cost(\sigma, \sigma', \lambda_{\sigma,\sigma'})$.

To obtain the distance between any two events, we project the normalized distance between traces onto the individual pairs of events. Formally, we construct an undirected weighted graph $G = (E_C, R, l, w)$ where nodes E_C are the events of $L = (C, l)$ with labeling l. For each pair (σ, σ') of traces in L and a best matching $\lambda_{\sigma,\sigma'}$ and for each pair $(e, e') \in \lambda_{\sigma,\sigma'}$ of events, we add the edge (e, e') to R with weight $w(e, e') = cost(\sigma, \sigma', \lambda_{\sigma,\sigma'})/maxCost$. Note that in G a single event may have many weighted edges describing how close it is to the most similar event in other traces. When mapping the costs from pairs of traces to pairs of events in G, any edge between events with a precise label a (i.e., $a \notin A'$) gets cost 0. This way, we will enforce that these labels are not refined. The higher the cost between two events, the more likely that they receive different labels. Searching for a mapping with least cost ensures we group the most similar events and give them together the same label during refinement. Figure 4(a) shows an example of a weighted graph of events for an imprecise log that consists of four traces. Note that the graph is incomplete; the mappings between t_0 and t_2, t_1 and t_3, and t_0 and t_3 are not shown for the sake of simplicity.

5.3 Refining Labels Horizontally Across Variants

We can now identify variants *within* a process by grouping events across traces based on their similarity. The reason for distinguishing variants is the following. If two very different variants of a part of the process are considered together, a

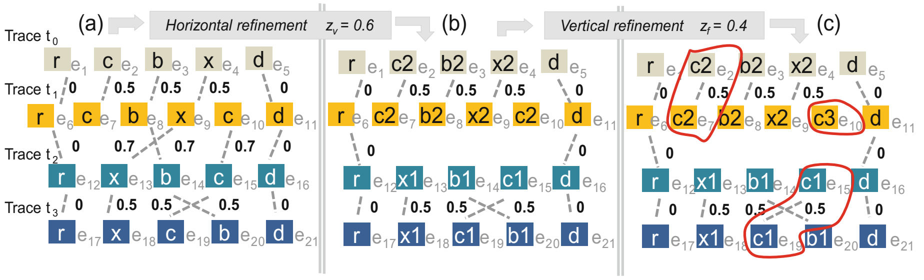
Trace t0
Trace t1
Trace t2
Trace t3
(a)
Horizontal refinement z_v = 0.6
(b)
Vertical refinement z_f = 0.4
(c)

Fig. 4. A graph of labeled events with weighted edges denoting the dissimilarity (a), for which the labels are refined horizontally (b) and then vertically (c).

discovery algorithm may return a more general structure than exists in reality. Consider for example the two traces $\sigma = \langle ..., c, b, x, ... \rangle$ and $\sigma' = \langle ..., x, b, c, ... \rangle$. One may consider them a single variant and return for example a model with activities b, c and x in parallel (i.e. can be executed in any order). However, an alternative would be having a precise model that only allows these two variants. The "optimal" model depends on the particular case. When the original model for the system is unknown, we cannot claim one of them is better, therefore we simply want to add the alternative with both variants to the solution space of existing discovery algorithms allowing user to explore both representations. Label refinement allows us to achieve this systematically.

The similarity measure enables us to be flexible when considering which variants to split by introducing a variant threshold z_v. We say, two traces σ and σ' are *in the same variant*, written $\sigma \sim \sigma'$ iff their normalized $cost(\sigma, \sigma', \lambda_{\sigma,\sigma'})/maxCost \leq z_v$ or there exists σ'' with $\sigma \sim \sigma'' \sim \sigma'$. Note that $\sim$ is an equivalence relation where two very dissimilar traces may become equivalent if there is a "chain" of similar traces between them. Thus, two events $e \in \sigma, e' \in \sigma'$ that have imprecise labels (i.e., $l(e), l(e') \in A'$) *are in the same variant* iff $\sigma \sim \sigma'$. In our graph G, we materialize (dis-)similarity by removing any edge (e, e') with weight $w(e, e') > z_v$. As all mappings between events of the same two traces σ and σ' carry the same weight, all events of a trace are kept in the same variant. Note that edges between the events that carry precise labels have weight 0 and are not split into multiple variants.

For example, setting the variant threshold for the event distance graph G of Fig. 4(a) to 0.6 yields the graph G' of Fig. 4(b) showing two variants in the part of the process involving labels c, b, x. Labels r and d are not refined into multiple variants.

5.4 Refining Labels Vertically Within Variant

After refining labels horizontally to distinguish different variants, there can still be multiple events carrying the same label within a single variant indicating either a loop or different tasks. Assuming in 50 % of cases activity c is executed once and in the other 50 % of the cases, c is executed twice, we could infer that there are two c tasks (one optional), or just one c task in a loop. In the following, we again use label refinement to add both alternatives to the solution space.

For refining labels within a single variant, we assume the following characteristics of a proper loop: when the number of iterations increases, the probability of executing this iteration decreases. For example, one may always execute the first iteration, whereas the second iteration is only executed in 20 % of the cases. In contrast, a duplicated task in a sequence would show similar numbers of executions in all traces of the same variant.

Based on this assumption, we introduce an *unfolding threshold* parameter z_f. For each imprecise label candidate $a \in A'$, let G_a^1, ..., G_a^m be the connected components of G in which all events have label a. G_a^i and G_a^j are in the same variant iff for any two events $e^i \in G_a^i$ and $e^j \in G_a^j$, e^i and e^j *are in the same variant* (see Sect. 5.3). For example, Fig. 4(c) highlights for imprecise label c the three connected components $G_c^1 = \{e_2, e_7\}$, $G_c^2 = \{e_{10}\}$, $G_c^3 = \{e_{15}, e_{19}\}$, in which G_c^1 and G_c^2 in the same variant. Next, let $\#G_a^i$ denote the average position of the events of $\#G_a^i$ in their respective traces. Let G_a^1 ... G_a^k be in the same variant ordered by $\#G_a^i$. Let $maxSize = max_{1 \leq i \leq k} \left|G_a^i\right|$ be the size of largest component (w.r.t. its events). For $1 \leq i \leq k$, if $i = 1$ or $\left|G_a^i\right| \geq v_f * maxSize$, then all events in G_a^i get a new label, otherwise G_a^i get the label of the events of G_a^{i-1}. For example, for imprecise label c, for the two connected components $G_c^1 = \{e_2, e_7\}$ and $G_c^2 = \{e_{10}\}$ that are in the same variant, $\#G_c^1 = 2$, $\#G_c^2 = 5$, and $maxSize = 2$. Therefore, if the *unfolding threshold* v_f is 0.6, then the events in G_c^2 get the same label as the events in G_c^1. If v_f is 0.4, then both G_c^1 and G_c^2 each get a new label.

6 Experimental Evaluation and Case Study

We implemented the techniques of Sect. 5 in the process mining toolkit ProM and conducted controlled experiments and a real-life case study to evaluate our approach. Plugins and experiments are available in the *TraceMatching* package of ProM. We first explain the experimental setup and then discuss the result.

Experimental Setup. The experimental setup is shown in Fig. 5. We randomly generated block structured models as systems with n number of visible tasks. Each system has k tasks that have the same activity label (here we consider just one duplicated label). For each system, we generate one imprecisely labeled log $L_{lab} = (C, l_{lab})$ of a 1000 cases each. From the imprecise log L_{lab}, we discover $M_{lab} = D(L_{lab})$. For the same log, we also apply our approach of Sect. 5 to obtain a refined log $L_{re} = (C, l_{re})$ (note that $\beta(L_{re}) = L_{lab}$), for which we discover model M_{re}. Two algorithms are used: IM [11], i.e., $M_{re,IM} = \beta(D_{IM}(L_{re}))$, and ILP [10], i.e., $M_{re,ILP} = \beta(D_{ILP}(L_{re}))$. The quality of each of the models is compared with the corresponding model M_{lab} for evaluating to what extent our aims has been achieved. In all experiments, the same cost configuration is used for matching events. To speed up the experiments, the events that have precise

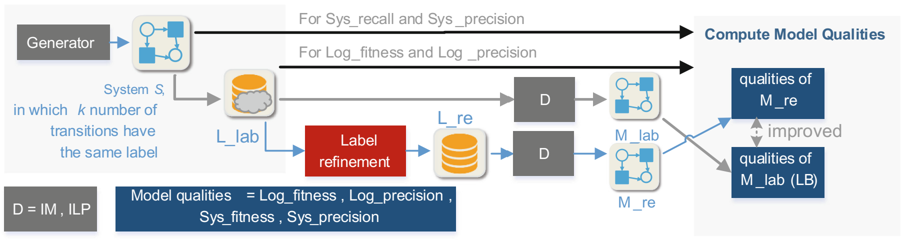
Generator
For Sys_recall and Sys_precision
For Log_fitness and Log_precision
Compute Model Qualities
System S_i in which k number of transitions have the same label
L_lab
Label refinement
L_re
D
D
M_lab
M_re
qualities of M_re
improved
qualities of M_lab (LB)
D = IM, ILP
Model qualities = Log_fitness, Log_precision, Sys_fitness, Sys_precision

Fig. 5. An overview of the experimental design.

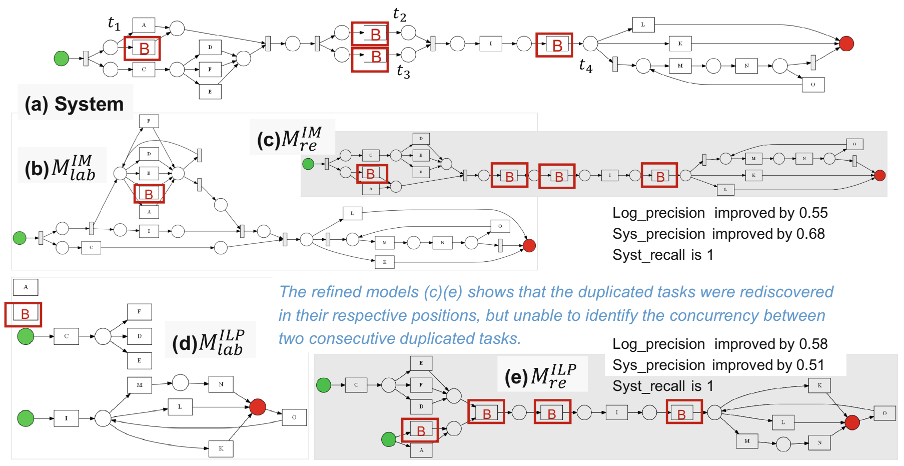
(a) System
(b) M^{IM}_{lab}
(c) M^{IM}_{re}
(d) M^{ILP}_{lab}
(e) M^{ILP}_{re}
Log_precision improved by 0.55
Sys_precision improved by 0.68
Syst_recall is 1
The refined models (c)(e) shows that the duplicated tasks were rediscovered in their respective positions, but unable to identify the concurrency between two consecutive duplicated tasks.
Log_precision improved by 0.58
Sys_precision improved by 0.51
Syst_recall is 1

Fig. 6. Original model with duplicate tasks (a), results of IM on imprecise log (b) and on refined log (c), same for ILP (d) and (e).

labels, i.e. $l(e) = l(e') \notin A'$, are matched naively based on their labels and ordering in their respective traces. All models, logs and results can be downloaded[2].

(Exp. 1) When Imprecise Labels are not in a Loop, What are the Improvements? In this experiment, we used the default parameters: the variant threshold z_v is 0.05 and the unfolding threshold z_f is 0.60 for all models. We generated for size $n = [10, 15, 20]$ 200 models (600 models in total). For each model, there are $k = 4$ transitions having the same label and that are not in a loop.

We show two examples of the refined models compared to their imprecise models in Figs. 6 and 7 to illustrate our results: Fig. 6 shows an improvement in *log_precision* of more than 0.50 and Fig. 7 an improvement of 0.10. In Fig. 6, the original model (a) has four duplicated tasks labeled "B"; applying IM and ILP on the imprecise log respectively results in discovering an imprecise model (b), which has a flower subprocess consisting of 5 activities, or an imprecise

[2] doi:10.4121/uuid:ea90c4be-64b6-4f4b-b27c-10ede28da6b6
or https://svn.win.tue.nl/repos/prom/Documentation/TraceMatching/BPM2016.zip.

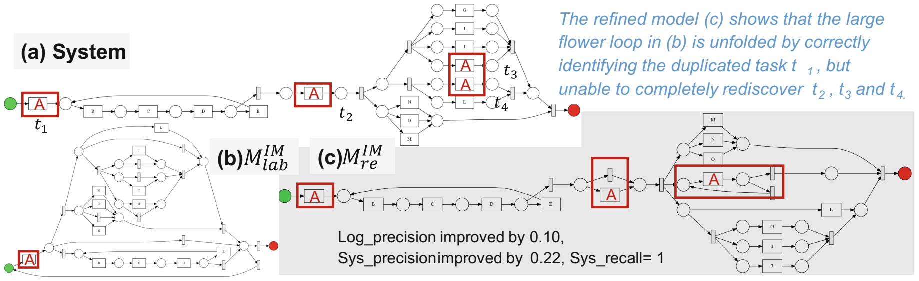
(a) System
t_1
t_2
t_3
t_4
The refined model (c) shows that the large flower loop in (b) is unfolded by correctly identifying the duplicated task t_1, but unable to completely rediscover t_2, t_3 and t_4.
(b)M^{IM}_{lab}
(c)M^{IM}_{re}
Log_precision improved by 0.10,
Sys_precision improved by 0.22, Sys_recall= 1

Fig. 7. Original model (a), result of IM on imprecise log (b) and on refined log (c).

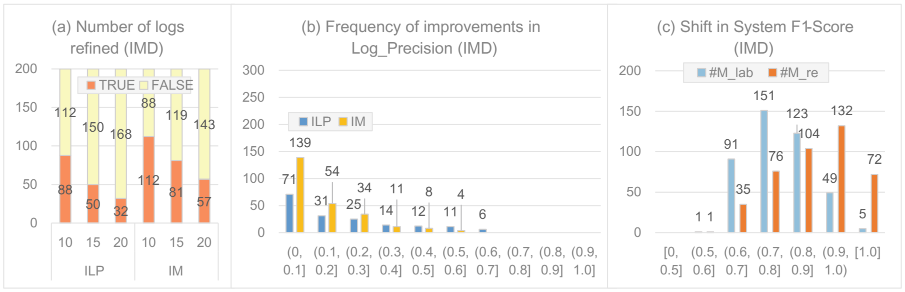

Fig. 8. Number of refined log (a), frequency of improvements in log precision (b) and shifts in system scores (c).

model (d), which has two unconnected activities; on the refined log, for both ILP and IM, the refined models (c) and (e) shows that the four duplicated tasks were correctly discovered in their respective positions in the process, however, our approach is unable to identify the concurrency between two consecutive duplicated tasks t_2 and and t_3 in (a).

Overall, Fig. 8(a) shows the number of systems for which our approach was able to find a refinement for its log that leads to discovering a better model with a higher log precision, while using automated detection of imprecise labels (IMD). In general, in 35 % (420 of 1200) of the logs, we were able to find a refinement with default parameters using IMD; using domain knowledge (OD) increased this number by 3 %. For 42 % of the refined logs, IM discovered an improved model, which is 14 % more than for ILP.

Figure 8(b) shows the histogram of frequencies of actual *log_precision* improvements using IMD. As can be seen, for both ILP and IM, our approach is able to help discover models with significant improvements. For ILP, the approach was able to find for 99 out of 600 models an improvement between 0.1 and 0.7 (using OD, this number increased by 9 %); similar for IM, 111 out of 600 refined models had such an improvement (using OD, this number is increased by 20 %). The average log precision is increased by 0.15.

Figure 8(c) shows the absolute $F1$-score (which is the harmony average of *sys_precision* and *sys_recall*) for M_{lab} (discovered on imprecise logs) versus M_{re} (discovered on the successfully refined logs using IM and ILP); our refinement clearly shifts the F1-score towards 1. When using automated detection (IMD), 16 % (67 out of 420) of the improved logs were refined in such way that $F1$-score becomes 1, which indicates that the resulting model have exactly the same alternative behavior as the original system enabled by the log (using OD we obtain 77 out of 516). Performance-wise, the average running time for computing one refined log varies between 8 and 14 s. depending on the model size.

(Exp. 2) Influence of Our Parameters. Next, we investigated whether adjusting parameters improves the quality of label refinement and whether such parameters can be found automatically for each model. For this, we repeated the above experiment for IM and OD and changed variant threshold (from 0.08 to 0.00 in steps of 0.01) and unfolding threshold (from 0.00 to 0.60 in steps of 0.10). We stopped when getting a log where M_{re} had higher log precision than M_{lab}. The average running time for computing one such refined log has increased to between 53 and 111 sec. depending on the model size.

Figure 9(a) shows the number of imprecise logs improved, (b) shows the actual improvements in *log_precision*, and (c) shows the F1-scores of *sys_recall* and *sys_precision*. It is worthwhile to note that, using the adaptive parameters, for 87 % of the imprecise logs, we were able to refine the log helping IM discover a better model. The average log precision is increased by 0.12. Compared to the

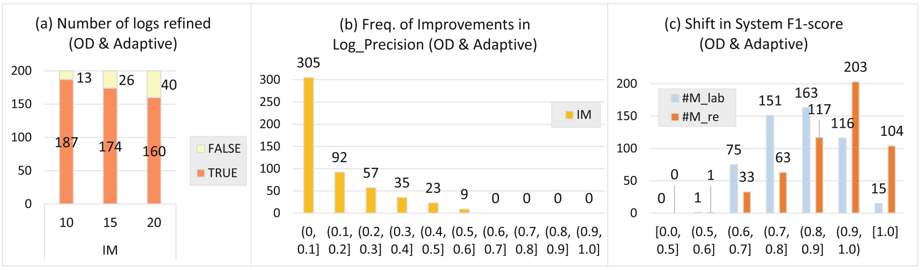

Fig. 9. The same types of result as Fig. 8 when using adaptive parameters.

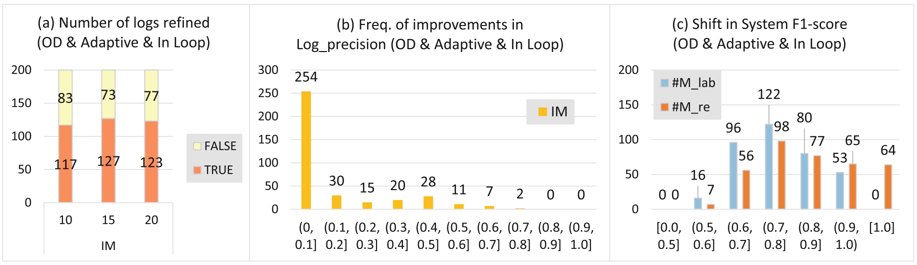

Fig. 10. The same types of result as Fig. 8, if a duplicated task is found in a loop.

46 % (when using default parameters and OD), the number is increased by more than 89 %. Another notable result is that the number of M_{re} that has an increase in *log_precision* between 0.2 and 0.7 is also increased by 72.2 % compared to the default parameter. This states for over one out of five logs, the adaptive approach is able to find a rather significant improvement, if the imprecise labels are not in a loop.

We manually inspected the models that could not be improved by using adjusted parameters. We found that this mostly concerned models that either have a large loop or have duplicated tasks concurrent to many other tasks. The difference in the corresponding components (i.e. such loops increase the cost of structure and such concurrency increases the cost of neighbors) becomes dominant in the cost returned by trace matching, resulting in splitting the imprecise labels wrongly even though the matching may be correct.

(Exp. 3) What if Imprecise Labels Appear in a Loop? We again generated 600 models, 200 for each $n = 10, 15, 20$. We used OD and set $k = 2$ transitions that have imprecise label: one inside and one outside of a loop. We used adaptive parameter selection and IM as discovery algorithm. Figure 10 shows the results. In 60.5 % of the models, the approach could find an improvement (32 % less compared to the results when no duplicated task is in a loop), which indicates that the approach has more difficulties to distinguish imprecise labels in loops. Another interesting result is that although the approach could improve fewer logs, the improvements achieved were considerable in some cases; 20 models have increased log precision by more than 0.5. Figure 11 shows an example of the model discovered using the refined log, which rediscovered the original model Fig. 11(a).

Inspecting the models, we observe that the approach is able to to distinguish loops if an imprecise transition t outside of a loop is followed by an imprecise transition inside of a loop. We found three patterns where our approach failed: (1) distinguishing a second iteration of a loop from a choice for a duplicate activity, (2) distinguishing a duplicate activity at the end of a loop body from one immediately after the loop, (3) one duplicate activity is concurrent to another duplicate activity within a loop. We plan to address these issues in our future work.

Real-Life Case Study. We conducted a case study involving a healthcare process. The log was provided by Maastricht University Medical Center (MUMC+), a large academic hospital in the Netherlands. We used existing approaches to filter the known deviating cases and events. The cleaned hospital log contains 1039 cases and 6213 events having five distinct labels. Since the log still contains imprecise labels and misses some events, applying the Inductive Visual Miner (IvM) yields an imprecise model with two self loops, as shown in Fig. 12(a). Using the default parameter, the approach was unable to refine the log. Therefore, we took an iterative approach.

We first refined events labeled with "surgery", i.e., the imprecise label candidate is "surgery". In the second and third iteration we refined events labeled with "consultation". The resulting model shows the sequential behavior expected

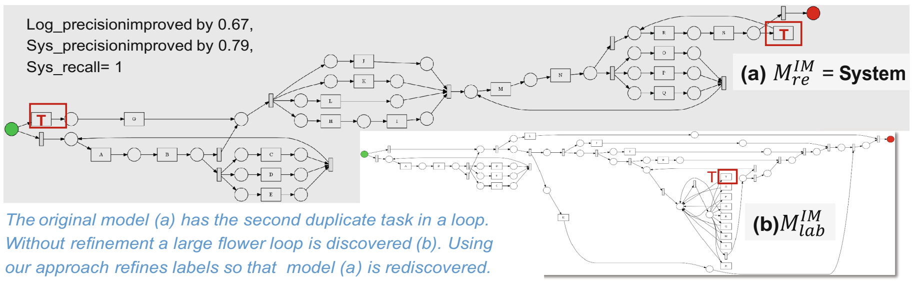
Log_precisionimproved by 0.67,
Sys_precisionimproved by 0.79,
Sys_recall= 1
(a) M_{re}^{IM} = System
(b)M_{lab}^{IM}
The original model (a) has the second duplicate task in a loop. Without refinement a large flower loop is discovered (b). Using our approach refines labels so that model (a) is rediscovered.

Fig. 11. Original model with duplicate tasks and rediscovered by IM on refined log (a), result of IM on imprecise log (b).

by domain experts. An interesting result is that after refining the labels, the discovered model is now suitable for computing performance. For example, a domain expert stated that within 2 months after the measurements, the first surgery should be executed, and the model shows on avg. 59 days. After the first surgery, a post-surgery consultation should take place within a week, and the model shows on avg. 8 days. If a second surgery should take place, then it should be performed after two weeks, and the model shows on avg. 14 days. Note that such performance diagnostics are difficult to obtain using the model discovered from the imprecise log Fig. 12(a).

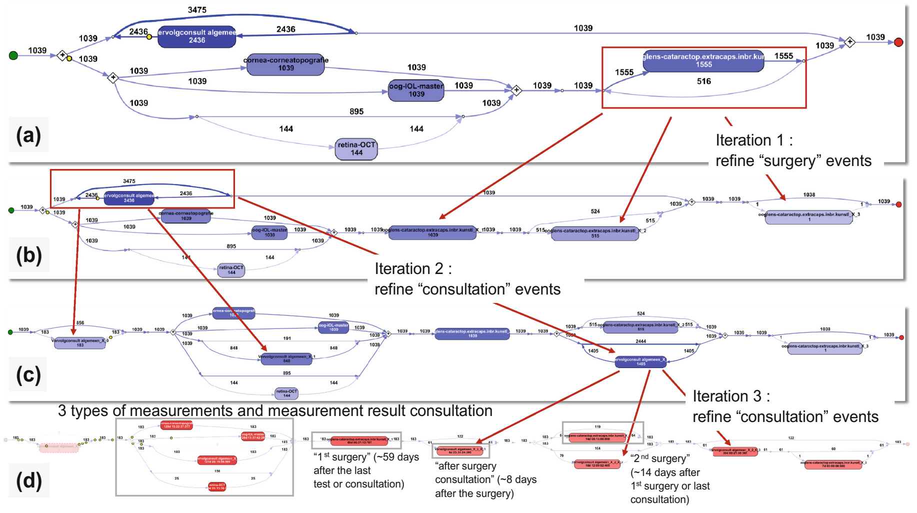
(a)
Iteration 1 : refine "surgery" events
(b)
Iteration 2 : refine "consultation" events
(c)
Iteration 3 : refine "consultation" events
3 types of measurements and measurement result consultation
(d)
"1st surgery" (~59 days after the last test or consultation)
"after surgery consultation" (~8 days after the surgery)
"2nd surgery" (~14 days after 1st surgery or last consultation)

Fig. 12. Real-life log obtained from a Dutch hospital that was refined our approach; the resulting model better reflects reality and can be used to diagnose performance.

7 Discussion and Conclusion

In this paper, we investigated the problem of imprecise labels and proposed a fresh look at the problem from a log preprocessing point of view. We used context and structural information of events in a log to find dissimilar groups of events that have the same label and refined their labels accordingly.

The results of our evaluation provide interesting insights. When imprecise labels are not in a loop, our approach is able to improve logs by refining labels in 35 % of the cases using a default parameter, which increased to 87 % if the parameter is automatically adapted to the log and the discovery algorithm. If one imprecise label is in a loop, we could still improve 61 % of the logs. The case study demonstrated that the approach can be used iteratively (i.e., refining labels in multiple steps) in practice to obtain more accurate and precise models. Interestingly, such a model can be used to derive reliable performance diagnostic. Future research aims at investigating and tackling the limitations of the approach found during the experiments.

References

1. Herbst, J.: A machine learning approach to workflow management. In: Proceedings 11th European Conference on Machine Learning, pp. 183–194 (2000)
2. van der Aalst, W.M.P., Rubin, V., Verbeek, H.M.W., van Dongen, B.F., Kindler, E., Günther, C.W.: Process mining: a two-step approach to balance between underfitting and overfitting. Soft. Syst. Model. **9**(1), 87–111 (2010)
3. De Weerdt, J., De Backer, M., Vanthienen, J., Baesens, B.: A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs. Inf. Syst. **37**(7), 654–676 (2012)
4. vanden Broucke, S.K.L.M.: Advances in process mining: artificial negative events and other techniques. Ph.D. thesis, KU Leuven (2014)
5. Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: Quality dimensions in process discovery: the importance of fitness, precision, generalization and simplicity. Int. J. Coop. Inf. Syst. **23**(1), 1–39 (2014)
6. Lu, X., Fahland, D., van den Biggelaar, F.J.H.M., van der Aalst, W.M.P.: Detecting deviating behaviors without models. In: Reichert, M., Reijers, H. (eds.) BPM Workshops 2015. LNBIP, vol. 256, pp. 126–139. Springer, Heidelberg (2016). doi:10.1007/978-3-319-42887-1_11
7. Pittke, F., Richetti, P.H.P., Mendling, J., Baião, F.A.: Context-sensitive textual recommendations for incomplete process model elements. In: BPM 2015, Proceedings, pp. 189–197 (2015)
8. Koschmider, A., Ullrich, M., Heine, A., Oberweis, A.: Revising the vocabulary of business process element labels. In: Zdravkovic, J., Kirikova, M., Johannesson, P. (eds.) CAiSE 2015. LNCS, vol. 9097, pp. 69–83. Springer, Heidelberg (2015)
9. de Medeiros, A.K.A., Weijters, A.J.M.M., van der Aalst, W.M.P.: Genetic process mining: an experimental evaluation. Data Min. Knowl. Discov. **14**(2), 245–304 (2007)
10. van Zelst, S.J., van Dongen, B.F., van der Aalst, W.M.P.: ILP-based process discovery using hybrid regions. In: Proceedings of the International Workshop on Algorithms and Theories for the Analysis of Event Data, ATAED 2015, pp. 47–61 (2015)

11. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs - a constructive approach. In: Colom, J.-M., Desel, J. (eds.) PETRI NETS 2013. LNCS, vol. 7927, pp. 311–329. Springer, Heidelberg (2013)
12. Greco, G., Guzzo, A., Pontieri, L., Saccà, D.: Discovering expressive process models by clustering log traces. IEEE Trans. Knowl. Data Eng. **18**(8), 1010–1027 (2006)
13. De Weerdt, J., vanden Broucke, S.K.L.M., Vanthienen, J., Baesens, B.: Active trace clustering for improved process discovery. IEEE Trans. Knowl. Data Eng. **25**(12), 2708–2720 (2013)
14. García-Bañuelos, L., Dumas, M., La Rosa, M., De Weerdt, J., Ekanayake, C.C.: Controlled automated discovery of collections of business process models. Inf. Syst. **46**, 85–101 (2014)
15. La Rosa, M., Dumas, M., Ekanayake, C.C., García-Bañuelos, L., Recker, J., ter Hofstede, A.H.M.: Detecting approximate clones in business process model repositories. Inf. Syst. **49**, 102–125 (2015)
16. van der Aalst, W.M.P., Adriansyah, A., van Dongen, B.F.: Replaying history on process models for conformance checking and performance analysis. Wiley Interdisc. Rev.: Data Min. Knowl. Discov. **2**(2), 182–192 (2012)
17. Munoz-Gama, J.: Conformance checking and diagnosis in process mining. Ph.D. thesis, Universitat Politècnica de Catalunya (2014)
18. van der Aalst, W.M.P., de Medeiros, A.K.A., Weijters, A.J.M.M.T.: Process equivalence: comparing two process models based on observed behavior. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 129–144. Springer, Heidelberg (2006)
19. van der Aalst, W.M.P., van Hee, K.M., ter Hofstede, A.H.M., Sidorova, N., Verbeek, H.M.W., Voorhoeve, M., Wynn, M.T.: Soundness of workflow nets: classification, decidability, and analysis. Formal Aspects Comput. **23**(3), 333–363 (2011)