HOSTED BY

ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/jides

Mining local process models

Niek Tax[a,b,*], *Natalia Sidorova*[a], *Reinder Haakma*[b], *Wil M.P. van der Aalst*[a]

[a] *Eindhoven University of Technology, P.O. Box 513, Eindhoven, The Netherlands*
[b] *Philips Research, Prof. Holstlaan 4, 5665 AA Eindhoven, The Netherlands*

HIGHLIGHTS

- This paper presents a new data mining method called local process model (LPM) mining.
- LPM mining extends sequential pattern mining techniques to more complex patterns.
- LPM mining enables process mining of noisy data by focusing on local structures.

ARTICLE INFO

Article history:
Published online 2 December 2016

Keywords:
Process mining
Knowledge discovery
Data mining

ABSTRACT

In this paper we describe a method to discover frequent behavioral patterns in event logs. We express these patterns as *local process models*. Local process model mining can be positioned in-between process discovery and episode/sequential pattern mining. The technique presented in this paper is able to learn behavioral patterns involving sequential composition, concurrency, choice and loop, like in process mining. However, we do not look at start-to-end models, which distinguishes our approach from process discovery and creates a link to episode/sequential pattern mining. We propose an incremental procedure for building local process models capturing frequent patterns based on so-called process trees. We propose five quality dimensions and corresponding metrics for local process models, given an event log. We show monotonicity properties for some quality dimensions, enabling a speedup of local process model discovery through pruning. We demonstrate through a real life case study that mining local patterns allows us to get insights in processes where regular start-to-end process discovery techniques are only able to learn unstructured, flower-like, models.

1. Introduction

Process mining aims to extract novel insight from event data [1]. Process discovery, the task of discovering a process model that is representative for the set of event sequences in terms of start-to-end behavior, i.e. from the start of a case till its termination, plays a prominent role in process mining. Many process discovery algorithms have been proposed and applied to a variety of real life cases (see [1] for an overview). A different perspective on mining patterns in event

Peer review under responsibility of Qassim University.
* *Corresponding author at: Eindhoven University of Technology, P.O. Box 513, Eindhoven, The Netherlands.*
E-mail address: n.tax@tue.nl (N. Tax).

http://dx.doi.org/10.1016/j.jides.2016.11.001

sequences can be found in the data mining field, where the episode mining [2] and sequential pattern mining [3] subfields focus on finding frequent patterns that are local, not necessarily describing the whole event sequences from start to end. Episode mining and sequential pattern mining have e.g. been used to analyze telecommunication networks [2], web navigational logs [2,4], and retail sales transactions [5].

Sequential pattern mining and episode mining are limited to the discovery of *sequential orderings* or *partially ordered sets* of events, while process discovery methods aim to discover a larger set of event relations, including sequential orderings, (exclusive) choice relations, concurrency, and loops, represented in process models such as Petri nets [6], BPMN [7], and process trees [8]. Process models that can be discovered with process discovery methods distinguish themselves from more traditional sequence mining methods like Hidden Markov Models [9] and Recurrent Neural Networks [10,11] in that process models can be visually represented and their visual representation can be used for communication between process stakeholders. However, process discovery is normally limited to the discovery of a model capturing the behavior of process instances as a whole, and not local patterns within instances. Our goal is to develop methods allowing to mine *local* process models positioned in-between simple patterns (e.g. subsequences) and start-to-end models. Local process models focus on a subset of the process activities and describe some behavioral pattern that occurs frequently within event sequences. Such local process models cannot be discovered by using standard techniques.

Imagine a sales department where multiple sales officers perform four types of activities: (A) register a call for bids, (B) investigate a call for bids from the business perspective, (C) investigate a call for bids from the legal perspective, and (D) decide on participation in the call for bid. The event sequences (Fig. 1(a)) contain the activities performed by one sales officer throughout the day. The sales officer works on different calls for bids and not necessarily performs all activities for a particular call himself. Applying discovery algorithms, like the Inductive Miner [12], yields models allowing for any sequence of events (Fig. 1(c)). Such "flower-like" models do not give any insight in typical behavioral patterns. When we apply any sequential pattern mining algorithm using a threshold of six occurrences, we obtain the seven length-three sequential patterns depicted in Fig. 1(d) (results obtained using the SPMF [14] implementation of the PrefixSpan algorithm [13]). However, the data contains a frequent non-sequential pattern where a sales officer first performs A, followed by B and a C in arbitrary order (Fig. 1(b)). This pattern cannot be found with existing process discovery and sequential pattern mining techniques. The two numbers shown in the transitions (i.e., rectangles) represent (1) the number of events of this type in the event log that fit this local process model and (2) the total number of events of this type in the event log. For example, 13 out of 19 events of type C in the event log fit transition C, which are indicated in bold in the log in Fig. 1(a). Underlined sequences indicate non-continuous instances, i.e. instances with non-fitting events in-between the events forming the instance of the local process model.

In this paper we describe a method to extract frequently occurring *local process models*, allowing for choice, concurrency, loops, and sequence relations. We leverage process trees [15] to search for local process models, and describe a way to recursively explore candidate process trees up to a certain model size. For convenience, we often use the Petri net representations for process trees. In fact, results can also be visualized as BPMN [7], EPC [16], UML activity diagram [17], or UML statechart diagram [17]. We define five quality dimensions that express the degree of representativeness of a local process model with regard to an event log: *support*, *confidence*, *language fit*, *coverage*, and *determinism*. Based on quality metrics, we describe monotonicity properties over some quality dimensions and show how they can be used to make the recursive search over process trees more efficient.

The paper is organized as follows. Section 2 introduces the basic concepts used in this paper. Section 3 describes related work in the fields of process discovery and sequential pattern mining. Section 4 describes our local process model mining approach. Section 5 introduces quality dimensions and metrics for local process models and discusses monotonicity properties. Section 6 describes a local process model evaluation approach based on alignments. Section 7 shows the relevance of the proposed technique using two real life data sets and compares the results with the results obtained with several related techniques. Section 8 concludes the paper.

2. Preliminaries

In this section we introduce process modeling notations, including Petri nets, process trees, which are used in later sections of this paper.

X^* denotes the set of all sequences over a set X and $\sigma = \langle a_1, a_2, \ldots, a_n \rangle$ a sequence of length n; $\langle\rangle$ is the empty sequence and $\sigma_1 \cdot \sigma_2$ is the concatenation of sequences σ_1, σ_2. $\sigma \restriction Q$ is the projection of σ on Q, e.g. $\langle a, b, c, a, b, c \rangle \restriction_{\{a,c\}} = \langle a, c, a, c \rangle$. $\#_a(\sigma)$ denotes the number of occurrences of element a in sequence σ, e.g. $\#_a(\langle a, b, c, a \rangle) = 2$.

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

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

We assume the set of all *process activities* Σ_L to be given. An *event* e is the occurrence of an activity $e \in \Sigma_L$. We call a sequence of events $t \in {\Sigma_L}^*$ a *trace*. An *event log* $L \in \mathbb{N}^{{\Sigma_L}^*}$ is a finite multiset of traces. For example, the event log $L = [\langle a, b, c \rangle^2, \langle b, a, c \rangle^3]$ consists of 2 occurrences of trace $\langle a, b, c \rangle$ and three occurrences of trace $\langle b, a, c \rangle$. We lift the sequence projection to the multisets of sequences in the standard way. For example, for the log L given above $L \restriction_{\{a,c\}} = [\langle a, c \rangle^5]$. We lift the number of occurrences in a sequence to multisets of sequences in the standard way, for example, $\#_a(L) = 5$.

Petri nets are directed bipartite graphs consisting of transitions and places, connected by arcs. Transitions represent activities, while places represent the enabling conditions of