Discovering high-quality process models despite data scarcity

Jan Niklas Adams[1,*], Jari Peeperkorn[2], Tobias Brockhoff[1], Isabelle Terrier[3], Heiko Göhner[3], Merih Seran Uysal[1], Seppe vanden Broucke[2,4], Jochen De Weerdt[2] and Wil M.P. van der Aalst[1]

[1]Chair of Process and Data Science, RWTH Aachen University, Aachen, Germany
[2]Research Center for Information Systems Engineering (LIRIS), KU Leuven, Leuven, Belgium
[3]Heidelberger Druckmaschinen AG, Heidelberg, Germany
[4]Department of Business Informatics and Operations Management, Ghent University, Ghent, Belgium

Abstract

Process discovery algorithms learn process models from executed activity sequences, describing concurrency, causality, and conflict. Concurrent activities require observing multiple permutations, increasing data requirements, especially for processes with concurrent subprocesses such as hierarchical, composite, or distributed processes. While process discovery algorithms traditionally use sequences of activities as input, recently introduced object-centric process discovery algorithms can use graphs of activities as input, encoding partial orders between activities. As such, they contain the concurrency information of many sequences in a single graph. In this paper, we address the research question of reducing process discovery data requirements when using object-centric event logs for process discovery. We classify different real-life processes according to the control-flow complexity within and between subprocesses and introduce an evaluation framework to assess process discovery algorithm quality of traditional and object-centric process discovery based on the sample size. We complement this with a large-scale production process case study. Our results show reduced data requirements, enabling the discovery of large, concurrent processes such as manufacturing with little data, previously infeasible with traditional process discovery. Our findings suggest that object-centric process mining could revolutionize process discovery in various sectors, including manufacturing and supply chains.

Keywords

Process Mining, Process Discovery, Object-Centric Process Mining

ER2023: Companion Proceedings of the 42nd International Conference on Conceptual Modeling: ER Forum, 7th SCME, Project Exhibitions, Posters and Demos, and Doctoral Consortium, November 06-09, 2023, Lisbon, Portugal
*Corresponding author.
niklas.adams@pads.rwth-aachen.de (J. N. Adams); jari.peeperkorn@kuleuven.be (J. Peeperkorn); brockhoff@pads.rwth-aachen.de (T. Brockhoff); uysal@pads.rwth-aachen.de (M. S. Uysal); seppe.vandenbroucke@kuleuven.be (S. vanden Broucke); jochen.deweerdt@kuleuven.be (J. De Weerdt); wvdaalst@pads.rwth-aachen.de (W. M.P. van der Aalst)
0000-0001-8954-4925 (J. N. Adams); 0000-0003-4644-4881 (J. Peeperkorn); 0000-0003-1115-6601 (M. S. Uysal); 0000-0002-8781-3906 (S. vanden Broucke); 0000-0001-6151-0504 (J. De Weerdt); 0000-0002-0955-6940 (W. M.P. van der Aalst)

1. Introduction

Throughout time, business endeavors have been supported by processes, ranging from bookkeeping to production processes. With the advent of modern information systems, data traces of process executions are recorded in the underlying databases [1]. While businesses mostly have some idea of how their processes look like, analyzing the recorded data allows them to uncover their real execution. Algorithms transforming the data of process executions, i.e., event logs, into process models are called process discovery algorithms [2].

A plethora of process discovery algorithms have been proposed over the last two decades [3]. Process discovery techniques assume the existence of an extracted event log composed of a set of event sequences (cases) describing different process executions. In general, these algorithms project each event sequence to its activity sequence and aim to uncover the sequentiality/causality, concurrency, or conflict between activities. While causality and conflict can be uncovered using relatively few activity sequences, concurrency needs to be confirmed using a large set of observations. For example, four concurrent activities could manifest in 24 different sequences, and seven concurrent activities already in 5040 possible sequences. The number of sequences grows factorial with the number of concurrent activities. Therefore, larger numbers of concurrent activities require more observations for process discovery. This becomes infeasible for large numbers of concurrent activities or event logs with very few cases.

Problems with highly concurrent behavior are especially present in processes that are composed of multiple, concurrently running and lightly interacting subprocesses. The amount of interaction between subprocesses is called the inter-object complexity. High inter-object complexity indicates a tight coupling between subprocesses, i.e., large amounts of shared control flow. Low inter-object complexity indicates concurrently running subprocesses. We depict an overview of typical processes and their inter-object complexity in Fig. 1. As inter-object complexity only captures control-flow complexity between subprocesses, we complement this dimension with intra-object complexity, capturing the control-flow complexity within subprocesses. We collect descriptions of typical real-life processes from the literature and depict their mapping onto these two dimensions in Fig. 1. Specifically, we show processes considered in traditional process mining (workflows like ticketing processes and business process case management [4, 5]), discrete manufacturing systems [6] (job shop/mass customization [7], (flexible) assembly lines [6]), and composite workflows (business process like P2P/O2C [8], end-to-end order processing [9]), and supply chains [10]. Additionally, we depict where a traditional, completely linear process would be positioned.

While traditional event logs record cases as sequences of events, Object-Centric Event Logs (OCELs) [11] encode cases as graphs of events [12], preserving partial orders between events. By extracting the event data of compound processes as OCELs instead of traditional event logs, the concurrency between the subprocesses can be preserved using these graph-based process executions. Object-centric process discovery was recently introduced [13], using an OCEL as input and discovering a model composed of multiple interacting subprocesses. This enables process owners to discover a process directly from

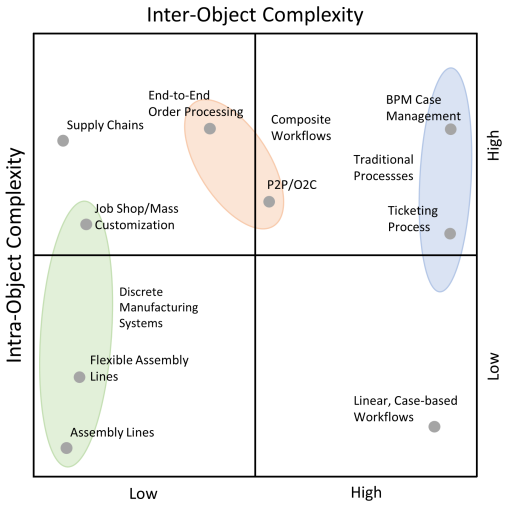

Figure 1: Categorization of different real-life processes based on inter- and intra-object complexity.

the graph-based process executions, rather than forcing them into the sequential structure needed for traditional process discovery and removing concurrency information.

In this paper, we address the following research question: How do data requirements reduce when using object-centric process discovery instead of traditional process discovery? Specifically, we want to break down the effect of employing object-centric discovery for different groups of processes with respect to their control-flow complexity between and within subprocesses, mapping the reduction in data requirements back to real-life processes.

We tackle these research questions through two main contributions: First, we formally define inter- and intra-object complexity and an evaluation framework that can exactly assess the discovered model quality based on the sample size of an event log. We instantiate this evaluation framework with 45.000 model and sample size combinations and compare the quality of the discovered model between traditional discovery and object-centric discovery. We map the results back to the dimension of inter- and inter-object complexity and assess which processes have the highest data requirement reduction when using object-centric process discovery. Second, we complement this experimental evaluation with a large-scale case study. Since the experimental evaluation is very computation heavy, it is only applicable to smaller models. Our case study shows the successful application of object-centric process discovery on a large real-life manufacturing process with hundreds of activities. It also shows that traditional process discovery fails to deliver the same quality.

2. Related work

Object-centric process mining addresses the problem of distinguishing multiple objects involved in a process. These objects can be used to identify subprocesses. In the past, this

problem has extensively been studied from the modeling side. Different process modeling languages/notations to capture object-centric processes have been proposed, such as artifact-centric process models [14], Object-Centric Behavioral Constraints (OCBC) [15], proclets [16], DB-Nets [17], COA-Nets [18], and t-PNIDs [19]. Some of these modeling techniques have been accompanied by a data format that is able to capture event data for processes of this kind. Artifact-centric event logs [20, 21], and eXtensible Object-Centric event logs XOC [22] have been proposed. Furthermore, Esser and Fahland have recently proposed a general-purpose graph database to store object-centric event data [23]. Object-centric process mining tackles the problem of object-centricity with the modeling language of object-centric Petri nets [13] and the accompanying data format of OCELs [11]. Both of these approaches have been developed to improve the complexity and scalability issues of existing modeling and data storage formats.

The core motivation of process discovery is the uncovering of as-is processes from real-life data. As such, process-discovery techniques aim to connect event data with process models. For the case of object-centric process mining, discovery techniques have been proposed for the modeling languages where data storage formats exist, namely artifact-centric discovery [20] and OCBC discovery from XOC [24]. Van der Aalst and Berti have introduced a general approach for discovering object-centric Petri nets from object-centric event data [13]. Subsequently, these process models are utilized in other data-driven process mining tasks, such as conformance checking [25] or enhancement [26]. This approach applies a traditional process discovery to each object type and merges the resulting models at interaction points. Any traditional process discovery algorithm can be employed in object-centric discovery.

Traditional process discovery describes the problem of mapping a set of activity sequences to a process model [2]. Many different techniques have been proposed [3], where the Split Miner [27] and the Inductive Miner [28] remain among the most popular algorithms. In our paper, we use the inductive miner due to its guarantee of sound workflow nets for every individual object type. This paper provides a quantitative and qualitative study as to how much process discovery results can be improved by individually discovering and merging process models for each object type rather than discovering one model on the flattened event data of all object types.

3. Traditional and object-centric process discovery

We introduce the foundations of object-centric and traditional event data and process discovery in this section. We link OCELs to traditional event logs and define the flattening of a traditional event log. Analogously, we introduce object-centric Petri nets and their relationship to traditional Petri nets.

3.1. Linking object-centric and traditional event data

First, we introduce some notations used throughout this paper. $\mathscr{E}$ is the universe of event identifiers, $\mathcal{OT}$ is the universe of object types, and $\mathcal{O}$ is the universe of objects. Each object is associated with exactly one object type through $\pi_{type} : \mathcal{O} \to \mathcal{OT}$. $\mathcal{T}$ denotes the universe

a) Object Associations of Object-Centric Process Mining

event	Activity	Time	Rim	Tire	Axle	Wheel	Mounted Axle
e_1	Mill Axle	15.06. 15:01			Axle1		
e_2	Press Tire Profile	15.06. 16:17		Tire1			
e_3	Press Tire Profile	15.06. 16:19		Tire2			
e_4	Mill Rim	16.06. 07:49	Rim1				
e_5	Drill Axle	16.06. 09:56			Axle1		
e_6	Add Valve	16.06. 13:01		Tire2			
e_7	Add Valve	16.06. 13:02		Tire1			
e_8	Mill Rim	18.06. 07:48	Rim2				
e_9	Construct Wheel	18.06. 08:12	Rim1	Tire2		Wheel1	
e_{10}	Construct Wheel	18.06. 08:15	Rim2	Tire1		Wheel2	
e_{11}	Mount Wheels to Axle	18.06. 10:04			Axle1	Wheel1, Wheel2	Mx1

b) Flattening: Case Notion of Traditional Process Mining

Case
Car1
Car1
Car1
Car1
Car1
Car1
Car1
Car1
Car1
Car1
Car1

Figure 2: An object-centric event log (a) and its flattened counterpart (b).

of event timestamps and $\mathscr{A}$ the universe of event activities. We denote the powerset of a set X, the set of all subsets, with $\mathscr{P}(X)$. A sequence of length $n \in \mathbb{N}$ orders elements of a set X and is denoted with $\sigma : \{1, \dots, n\} \to X$ and $\sigma = \langle x_1, \dots, x_n \rangle$. The set of directly follows relationships in a sequence is denoted by $df(\langle x_1, \dots, x_n \rangle) = \{(x_i, x_i + 1) \mid i \in \{1, \dots, n-1\}\}$.

Definition 1 (Object-Centric Event Log). An object-centric event log is a tuple $L = (E, O, OT, \pi_{act}, \pi_{time}, \pi_{trace})$ consisting of

- events $E \subseteq \mathscr{E}$, objects $O \subseteq \mathscr{O}$, object types $OT = \{\pi_{type}(o) \mid o \in O\}$,
- activities $\pi_{act} : E \to \mathscr{A}$, timestamps $\pi_{time} : E \to \mathscr{T}$, and
- ordering $\pi_{trace} : O \to E^*$ mapping each object to a sequence of events such that $\forall_{o \in O}\, \pi_{trace}(o) = \langle e_1, \dots, e_n \rangle \wedge \forall_{i \in \{1,\dots,n-1\}}\, \pi_{time}(e_i) \leq \pi_{time}(e_{i+1})$

Each event is linked to objects $\pi_{obj}(e) = \{o \in O \mid e \in \pi_{trace}(o)\}$.

An example of an OCEL is depicted in Fig. 2. Each row is an event associated with objects of different types, an activity, and a timestamp. Each object is associated with a sequence of events, e.g., $\pi_{trace}(Tire2) = \langle e_2, e_7, e_{10} \rangle$.

A traditional event log is a special case of an OCEL where objects are all of the same type and an event is associated with exactly one object.

Definition 2 (Traditional Event Log). An object-centric event log $L=(E, O, OT, \pi_{act}, \pi_{time}, \pi_{trace})$ is a traditional event log iff $|OT| = 1 \wedge \forall_{e \in E} |\pi_{obj}(e)| = 1$.

An event log consists of different executions of the same process. Each execution contains multiple events. In traditional process mining, a process execution is called a case and is a sequence of events for one object. We generalize this notion such that a process execution is a graph of events for multiple, connected objects [12]. The graph describes a partial order of events induced by the precedence constraints of each π_{trace} of the involved objects.

Definition 3 (Process Executions [12]). Let $L = (E, O, OT, \pi_{act}, \pi_{time}, \pi_{trace})$ be an object-centric event log. The object graph of object co-appearances is defined by $OG_L = (O, I)$

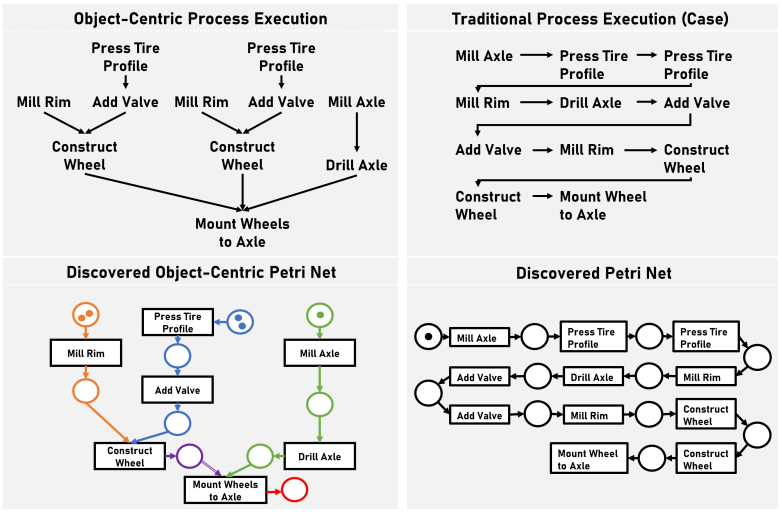

Figure 3: Top: A graph-based process execution from the object-centric event log of Fig. 2 a) and the sequential process execution from the flattened, traditional event log of Fig. 2 b). Bottom: The process models discovered from both of these process executions.

with $I = \{\{o, o'\} \mid o \neq o' \wedge \exists_{e \in E} \{o, o'\} \subseteq \pi_{obj}(e)\}$. The set of interdependent object sets are the connected components of the object graph $cc(L) = \{O' \subseteq O \mid O' \text{ form a connected component in } OG_L\}$. One set of dependent objects $X \in cc(L)$ spans a process execution. A process execution is a graph $p_X = (E_X, D_X)$ of nodes $E_X = \{e \in E \mid \pi_{obj}(e) \cap X \neq \emptyset\}$ and edges $D_X = \{(e, e') \in E_X \times E_X \mid \exists_{o \in X} (e, e') \in df(\pi_{trace}(o))\}$. The set of all process executions of an event log is defined by $px(L) = \{p_X \mid X \in cc(L)\}$.

The upper left part of Fig. 3 contains an example of the object-centric process execution contained in Fig. 2. The process execution shows the production of different parts that run concurrently and are assembled in a bottom-up way.

An OCEL can be transformed into the traditional event log format by flattening [11] it. This involves the choice of a case notion and the sequentializing of events for each object of that case notion. The case notion can be chosen freely, typically either a single object type is chosen or a set of connected objects. Flattening transforms a graph-based structure of process executions into a less expressive sequential structure of process execution, therefore, some information loss will never be preventable. We choose to flatten with all connected objects, i.e., flattening a process execution from a graph into a sequence. By doing this, we prevent events from disappearing or being duplicated as it would be the case if one would flatten on a single object type [11].

Definition 4 (Flattening). Let $L = (E, O, OT, \pi_{act}, \pi_{time}, \pi_{trace})$ be an object-centric event log. $L^{flat} = (E, O', OT', \pi_{act}, \pi_{time}, \pi'_{trace})$ is the flattened event log with three modified elements:

- a new, single object type $OT' = \{ot\}$ for an $ot \in \mathcal{OT} \wedge ot \notin OT$.
- new object identifiers $O' \subseteq \{o \in \mathcal{O} \mid \pi_{type}(o) \in OT'\}$ associated with the objects of the process executions $cc(L)$ through a bijection $\pi_{flat} : O' \to cc(L)$.

- event sequences for the new objects $\pi'_{trace}(o') = \langle e_1, \dots, e_n \rangle$ with $\{e_1, \dots, e_n\} = \bigcup_{o \in \pi_{flat}(o')} \pi_{trace}(o)$ and $\pi_{time}(e_1) \leq \dots \leq \pi_{time}(e_n)$ for $o' \in O'$.

We show the flattened event log in Fig. 2 b). The corresponding process execution, a sequence, is depicted in the upper right of Fig. 3. The concurrency information from the graph-based process execution is lost when flattening to a sequence.

3.2. Object-centric process models

A process can be represented as a process model, typically a Petri net. Object-centric processes are represented using an object-centric Petri net [13], a Petri net with places of different types and variable arcs which are able to consume more than one token.

Definition 5 (Object-Centric Petri Net). An object-centric Petri net is a tuple $OCPN = (P, \pi_{pt}, F_{var})$ of

- a Petri net $P = (T, P, F, l)$ consisting of transitions T, places P, arcs $F \subseteq (T \times P) \cup (P \times T)$, and a labeling function $l : T \nrightarrow \mathcal{A}$,
- a type function mapping each place to an object type $\pi_{pt} : P \rightarrow \mathcal{OT}$, and
- a set of variable arcs $F_{var} \subseteq F$.

We define the following notations for object-centric Petri nets:

- the preset of a transition $\bullet t = \{p \in P \mid (p, t) \in F\}$ for $t \in T$.
- the postset of a transition $t \bullet = \{p \in P \mid (t, p) \in F\}$ for $t \in T$.
- $tpl(t) = \{\pi_{pt}(p) \mid p \in \bullet t \cup t \bullet\}$ are the object types associated to transition $t \in T$.
- $tpl_{nv}(t) = \{\pi_{pt}(p) \mid p \in P \wedge \{(p, t), (t, p)\} \cap (F \setminus F_{\text{var}}) \neq \emptyset\}$ are the object types with non-variable arcs associated to transition $t \in T$.

An example of an object-centric Petri net is depicted in the lower left of Fig. 3. Places of different types are depicted with different colors, variable arcs are depicted with double lines. Please note that we dropped the output places of an object type's last transition for presentation purposes.

Analogously to object-centric and traditional event logs, a traditional Petri net is a special case of an object-centric Petri net where all places have the same type and no variable arcs exist. The bottom right of Fig. 3 depicts a traditional Petri net.

Definition 6 (Traditional Petri Net). An object-centric Petri net $OCPN = (P, \pi_{pt}, F_{var})$ is a traditional Petri net iff $|range(\pi_{pt})| = 1 \wedge F_{var} = \emptyset$.

We describe the state of an object-centric Petri net with a marking.

Definition 7 (Marking). Let $OCPN = ((T, P, F, l), \pi_{pt}, F_{var})$ be an object-centric Petri net. A token associates an object with a place. The set of tokens is define by $Q_{OCPN} = \{(p, o) \in P \times \mathcal{O} \mid \pi_{pt}(p) = \pi_{type}(o)\}$. A marking is a multiset of tokens $M \in \mathcal{B}(Q_{OCPN})$.

Given a marking, a transition can be enabled. If it is enabled, it can be executed. The execution of a transition is bound to objects. These are consumed in the input places and produced in the output places.

Definition 8 (Binding Execution). Let $OCPN = ((T, P, F, l), \pi_{pt}, F_{var})$ be an object-centric Petri net. $B_{OCPN}=\{(t,b) \in T\times(\mathcal{OT} \nrightarrow \mathcal{P}(\mathcal{O})) \mid dom(b)=tpl(t) \wedge \forall_{ot\in tpl_{nv}(t)}|b(ot)|=1\}$ defines the set of all possible bindings. The consumed tokens of a binding $(t,b)\in B_{OCPN}$ are defined by $cons(t,b)=[(p,o)\in Q_{OCPN} \mid p\in \bullet t \wedge o\in b(\pi_{pt}(p))]$ and the produced tokens are defined by $prod(t,b)=[(p,o)\in Q_{OCPN} \mid p\in t\bullet \wedge o\in b(\pi_{pt}(p))]$. A binding $(t,b)\in B$ in marking $M \in \mathcal{B}(Q_{OCPN})$ is enabled if $cons(t,b) \leq M$. Executing an enabled binding in M leads to marking $M'=M - cons(t,b) + prod(t,b)$. Executing an enabled binding $(t,b) \in B_{OCPN}$ in marking M is denoted with $M \xrightarrow{(t,b)} M'$. Multiple subsequent enabled bindings can be encoded as a binding sequence $\sigma=\langle(t_1,b_1),\dots,(t_n,b_n)\rangle\in B^*_{OCPN}$. A sequence that starts in marking M_0 and results in marking M_n is encoded as $M_0 \xrightarrow{\sigma} M_n$.

To define a process model that allows certain behavior, we need start and end points. These are sets of marking for an object-centric Petri net.

Definition 9 (Accepting Object-Centric Petri Net). Let $OCPN = ((T, P, F, l), \pi_{pt}, F_{var})$ be an object-centric Petri net and let $M_{init}, M_{final} \subseteq \mathcal{B}(Q_{OCPN})$ be initial and final markings of the object-centric Petri net. $OCPN_A = (OCPN, M_{init}, M_{final})$ is an accepting object-centric Petri net.

All binding sequences that lead from an initial marking to a final marking define the possible end-to-end behavior of the accepting object-centric Petri net. We define the language of the accepting object-centric Petri net as the set of all possible visible loop-free label sequences in the end-to-end behavior of the object-centric Petri net. In this way, we exclude binding sequences with loops that would render the language of the model infinite.

Definition 10 (Petri Net Language). Let $OCPN_A = (OCPN, M_{init}, M_{final})$ be an accepting object-centric Petri net. All loop-free valid binding sequences of the object-centric Petri net are given by $S(OCPN_A) = \{\langle(t_1,b_1),\dots,(t_n,b_n)\rangle \in B^*_{OCPN} \mid \exists_{M_i\in M_{init}} \exists_{M_f\in M_{final}} M_i \xrightarrow{(t_1,b_1)} \cdots \xrightarrow{(t_n,b_n)} M_f \wedge \forall_{i,j\in\{1,\dots,n\}\wedge i\neq j} M_i \neq M_j\}$. The language of the object-centric Petri net is the set of visible transition firing sequences $\Sigma(OCPN_A) = \{\langle l(t_1),\dots,l(t_n)\rangle \mid \langle(t_1,b_1),\dots,(t_n,b_n)\rangle \in S(OCPN_A)\}$

An accepting object-centric Petri net can be discovered from an OCEL using the general approach introduced by van der Aalst and Berti [13]. A Petri net is discovered for each object type. Given the cardinalities of different objects, the individual Petri nets are merged into one object-centric Petri net, connecting the individual nets at interconnecting transitions (transitions including multiple object types). Place types and variable arcs are assigned according to the types and cardinalities stemming from the individual subnets.

Definition 11 (Process Discovery). Let $L = (E, OT, O, \pi_{act}, \pi_{time}, \pi_{trace})$ be an object-centric event log. A process discovery algorithm $d(L) = OCPN_A$ returns an accepting object-centric Petri net for the object-centric event log.

This general approach discovers a traditional Petri net if a traditional event log is the input. The object-centric Petri nets in Fig. 3 are discovered from the OCEL and the flattened event log of Fig. 2.

3.3. Inter- and intra-object complexity

In this section, we formally define the inter- and intra-object complexity which were already informally introduced in the introduction. We later use these dimensions in our experimental framework to differentiate which process characteristics and, therefore, which real-life processes benefit most from employing object-centric discovery.

Definition 12 (Inter-Object Complexity). Let $OCPN_A = (((T, P, F, l), \pi_{pt}, F_{var}), M_{init}, M_{final})$ be an accepting object-centric Petri net. We define the following model attributes:

- $numt(OCPN_A) = |\{t \in T \mid t \in dom(l)\}|$ the number of non-silent transitions,
- $numot(OCPN_A) = |range(\pi_{pt})|$ the number of object types, and
- $inter(OCPN_A) = \frac{1}{max(1,|OT|-1)} \frac{\sum_{t \in T} |\{\pi_{pt}(p) | p \in \bullet t\}|-1}{|T|}$ is the inter-object complexity of the model.

The inter-object complexity is defined as the amount of shared transitions between objects. The more transitions are shared between objects, the more the object's control flows are interwoven with each other. Based on the definition, we make two observations.

Observation 13. High inter-object complexities point to many shared transitions. In the extreme case, all objects share all transitions. This would make objects redundant, as they all describe the same control flow. Only one object is necessary to encode the control flow, i.e., a process with an inter-object complexity of 1 is equivalent to a traditional process described by a Petri net.

Observation 14. A traditional process has an inter-object complexity of 1, as the single object of the process is involved in all transitions.

The inter-object complexity only describes the complexity of control flow between objects. To fully capture the complexity of the control flow, we further define the intra-object-complexity, capturing the control flow within objects.

Definition 15 (Intra-Object Complexity). Let $OCPN_A = (((T, P, F, l), \pi_{pt}, F_{var}), M_{init}, M_{final})$ be an accepting object-centric Petri net. For an object type $ot \in range(\pi_{pt})$, we retrieve the subnet $OCPN_{A,ot} = (P_{ot}, T_{ot}, F_{ot}, l_{ot})$ with

- $P_{ot} = \{p \in P \mid \pi_{pt}(p) = ot\}$,
- $T_{ot} = \{t \in T \mid \exists_{p \in \bullet t \cup t \bullet}\ \pi_{pt}(p) = ot\}$,
- $F_{ot} = \{(s,t) \in F \mid (s \in P_{ot} \wedge t \in T_{ot}) \vee (s \in T_{ot} \wedge t \in P_{ot})\}$, and
- $l_{ot}(t) = l(t)$ for $t \in T_{ot}$.

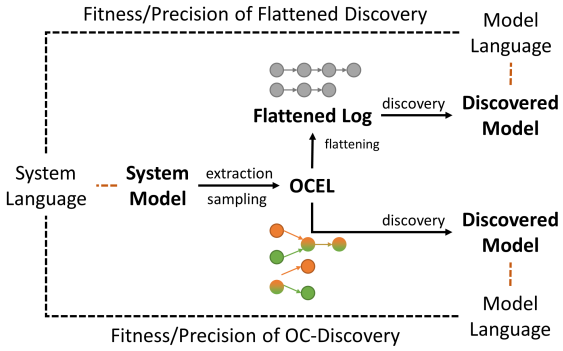

Figure 4: Experimental framework. We generate many different system models with varying characteristics.

We define the intra-object complexity for an individual type as $tioc(OCPN_{A,ot}) = \frac{|\Sigma(OCPN_{A,ot})|}{numt(OCPN_{A,ot})!}$. The intra-object complexity of the model is defined as the average intra-object complexity of all object types $intra(OCPN_A) = \frac{1}{numot(OCPN_A)} \cdot \sum_{ot \in range(\pi_{pt})} tioc(OCPN_{A,ot})$.

The intra-object complexity quantifies the objects' average deviation from a completely linear control flow. An intra-object complexity of 0 means that all objects follow a strictly linear control flow, while an intra-object complexity of 1 means that all objects would have a completely concurrent control flow.

4. Experimental framework

This section introduces the experimental setup to compare process discovery on object-centric and flattened event data which is depicted in Fig. 4. The experimental framework consists of three steps: Model generation, event log sampling, and discovery quality assessment.

Model generation We use a process model called the system model resembling a ground truth process. To cover a wide range of combinations of inter- and intra-object complexity, we generate a large set of different models. We depict the distribution of our generated system model along with some exemplary models in Fig. 5. In total, we have randomly generated 44570 system models with 6-8 visible activities (additionally, they can have silent transitions to model AND constructs) and two object types. These models do not have loops, as this would conflict with the next part of the framework, the event log sampling. We address this limitation in the threats to validity section at the end of Sec. 5. The model sizes are limited to maximum 8 visible transitions, as larger models would produce so many possible traces that an exact computation of the model language, as described in the next section, would become infeasible. However, we complement the

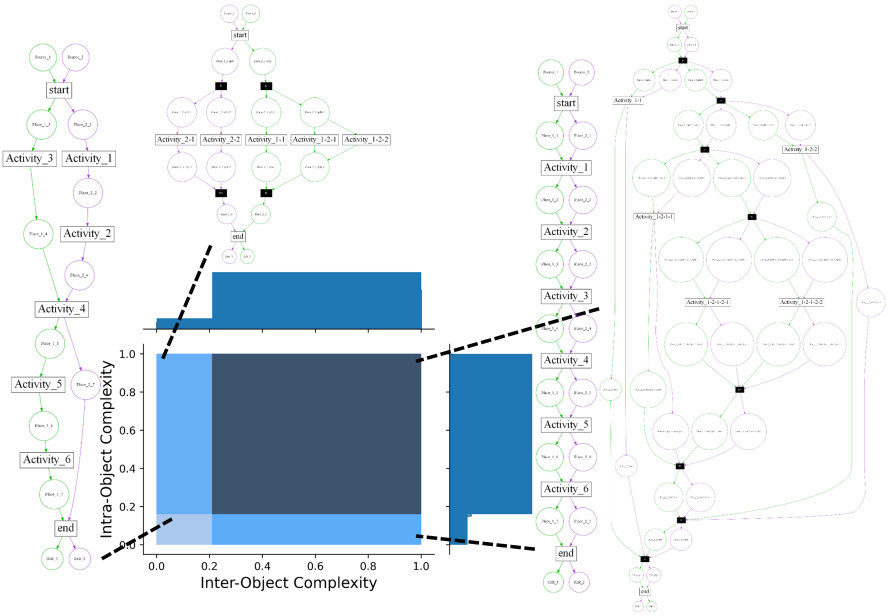

Figure 5: Distribution of generated model characteristics and example models for different combinations.

experiments with a case study on a large production process, showing the applicability and improvements of discovery also for very large models.

Event log sampling We extract an event log by sampling from the language of the system model. To do so, we compute the language of the system model, i.e., all possible process executions. We sample a subset of this language to simulate extracting an event log that only contains a part of the possible behavior.

Definition 16 (Event Log Extraction). Let $OCPN_A$=$(OCPN, M_{init}, M_{final})$ be an accepting object-centric Petri net and $s \in [0,1]$ be an extraction sampling rate. $sample(OCPN_A, s) \subseteq \Sigma(OCPN_A)$ samples a subset of the Petri net language corresponding to the sampling rate s. The language subset is mapped to an OCEL by $gen(sample(OCPN_A, s), OCPN_A) = (E, O, OT, \pi_{act}, \pi_{time}, \pi_{trace})$.

The presented approach allows us to quantify a sampling rate, i.e., how much of the possible model behavior is contained in the event log. Since we computed the full system model language, we can also compare the language of a discovered model to the language of a system model, quantifying how well the discovered model corresponds to the ground-truth system model given the sampling rate. This is explained in the next paragraph

Quantifying discovered model quality We compare the quality of discovered process models from the object-centric and flattened event logs. Starting from the OCEL, we discover a model before and after flattening the event log using Inductive Miner [28].

The model quality is defined as the recall/precision of the discovered model language with respect to the language of the system model. When the model is discovered from a sampled event log of the system model, the fitness defined here also measures the generalization capabilities of the discovery technique [29].

Definition 17 (Model Quality). Let $OCPN_A = (OCPN, M_{init}, M_{final})$ be a system model and $s \in [0, 1]$ be a sampling rate. The sampled OCEL is given by $L = d(gen(sample(OCPN_A, s), OCPN_A)$. The discovered model on the object-centric event log is denoted with $OCPN_A^d = d(L)$ and the discovered model on the flattened event log is denoted with $OCPN_A^{flat,d} = d(L^{flat})$. For any of the two discovered models $PN \in \{OCPN^d, OCPN^{flat,d}\}$, we define the fitness $fit(OCPN_A, PN) = \frac{|\Sigma(OCPN_A) \cap \Sigma(PN)|}{|\Sigma_d(OCPN_A)|}$ and the precision $prec(OCPN_A, PN) = \frac{|\Sigma(OCPN_A) \cap \Sigma(PN)|}{|\Sigma(PN)|}$.

Using this setup, we retrieve the discovered model quality for traditional and object-centric process discovery for different combinations of inter-object complexity, intra-object complexity, and sample sizes.

5. Experimental results

We present our experimental results in this section. The implementation is publicly available[1]. We present the dependency between discovered model quality and event log sample size for traditional and object-centric discovery on system models with varying inter- and intra-object complexity, as described in Sec. 4. The results are depicted in Fig. 6 (fitness) and Fig. 7 (precision).

We split the models according to their characteristics using low/high inter- and intra-object complexity. The decision for low and high values are made according to Fig. 5, i.e., an inter-object complexity of more than 0.2 is considered high and an intra-object complexity of less than 0.15 is considered to be low. We choose both values for the following reasons: First, high inter-object complexity indicates high redundancies between object types, i.e., the system model is behaviorally very close to a traditional Petri net. By choosing a low threshold we single out system models that are significantly different from traditional process models. Second, high intra-object complexity models indicate a lack of structure in the process, i.e., no order is enforced. By choosing a low threshold, we can single out system models that show high degrees of sequentiality in one bin and less structured models in the other bin.

The highest quality differences can be observed when the system model exhibits low levels of inter-object complexity, especially when the intra-object complexity is also low, i.e., each object has a relatively sequential path and only some interaction points with other objects. In this case, the average fitness of the discovered process model for extracted OCELs with a low sample rate is already 0.9. The fitness of the discovered model from the flattened event log is very low, starting at almost 0. Mapping this back to the initial taxonomy depicted in Fig. 1, object-centric discovery can significantly

[1] https://github.com/jn-adams/ObjectCentricDiscovery

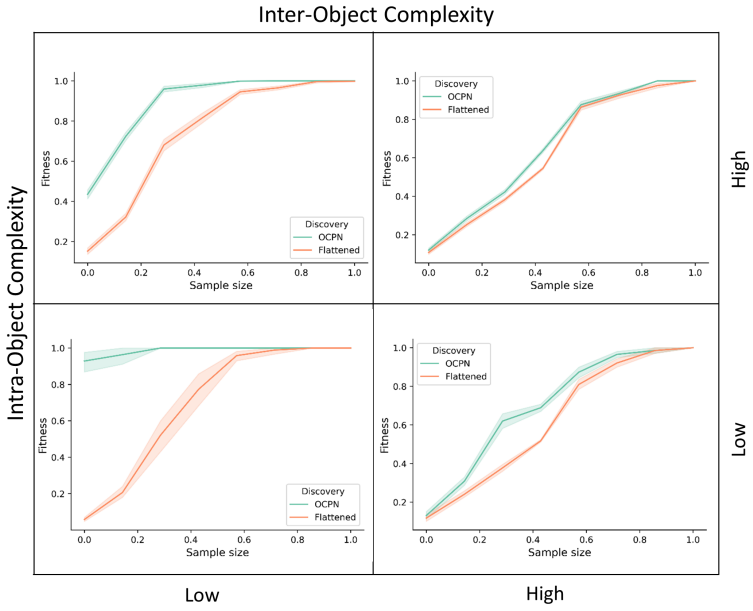

Figure 6: Discovered model fitness depending on the sample size of the event log for the four different quadrants of our process taxonomy.

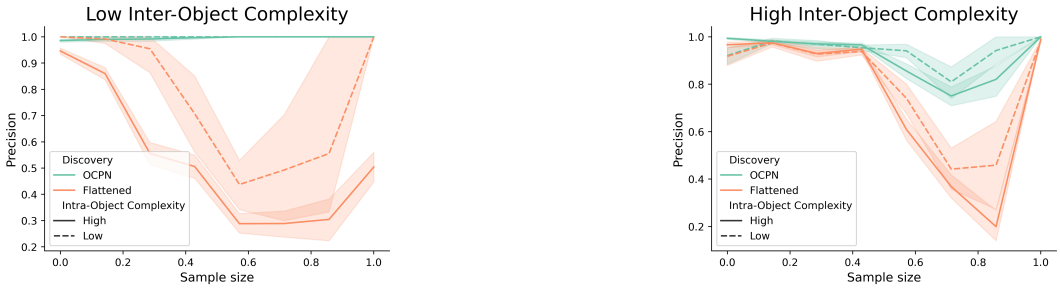

Figure 7: Discovered model precision depending on the sample size of the event log for the four different quadrants of our process taxonomy.

reduce the data required for manufacturing processes, supply chains, and composite workflows. Especially for processes with relatively sequential subprocesses, like assembly lines, object-centric discovery allows for discovery with much less data compared to traditional discovery. This means, that discovery for larger models, which was prohibitive due to the data requirements in traditional discovery, is now enabled using object-centric discovery. We will also see this in the case study in Sec. 6. Models with high inter-object complexity cannot benefit that much from switching to object-centric discovery. Since high inter-object complexity points to redundant control flows and is, in the extreme case, equivalent to traditional process models, these results are not surprising. It is notable, that – within this evaluation – object-centric discovery does not worsen results.

When considering the precision of discovered models, the quality differences are larger

for models with high intra-object complexity, although precision levels for low intra-object complexity are, generally, higher. This can be explained by low levels of intra-object complexity allowing more concurrent behavior, such that the discovery of flower-like models will be more precise than for more restrictive system models. In general, we observe that the high amounts of sequences produced by concurrent behavior push the discovery algorithm to discover flower models for the flattened event logs, significantly reducing the precision.

5.1. Threats to validity

As our experimental framework computes the exact languages of the models, it is computationally quite demanding, leading us to restrict the system model generation in two major ways: without loops and only limited to eight visible transitions. These are also the main limitations of this experimental framework: First, our results cannot be generalized to process models containing many loops. Second, our generated models only cover relatively small models of 8 visible transitions. While the results are already very clear and the difference should only increase for larger models, our experiments cannot prove this claim. Therefore, we complement the experiments with a case study performing object-centric discovery on a large-scale production process and comparing the results to traditional process discovery. We aim to mitigate the limitations of our experimental framework using this case study, as it shows that object-centric discovery is feasible for large processes and provides better results than traditional discovery.

6. Case study

We use object-centric event data from a production process in the German manufacturing company Heidelberger Druckmaschinen AG [30]. The confidential original event log contains hundreds of process executions with more than 800 activities. Object-centric discovery can be applied to the original event log, however, the resulting visualization is too large for human comprehension. Therefore, we limit our analysis to a subprocess of 105 activities. Our sublog contains 13 process executions. We, first, apply object-centric process discovery and, second, flatten the event log and apply traditional process discovery. Even though we do not know the true ground truth process to provide an exact sample rate, it would be close to 0 given the large number of concurrent activities and the low number of process executions. The results are depicted in Fig. 8.

The structure of the process can be rediscovered using object-centric discovery: Individual concurrent object paths are visible, ending up in one assembly activity that combines individual components into an output component. This output component follows its individual path afterward. The low number of process executions is not an issue when using object-centric discovery, it can still rediscover the concurrency between objects. This is not the case for process discovery on flattened event data. The flattened process discovery algorithm shows a flower model, i.e., the structure of the process is completely lost. Furthermore, it is generally infeasible to expect that a flattened event log would contain enough process executions such that concurrency could completely be

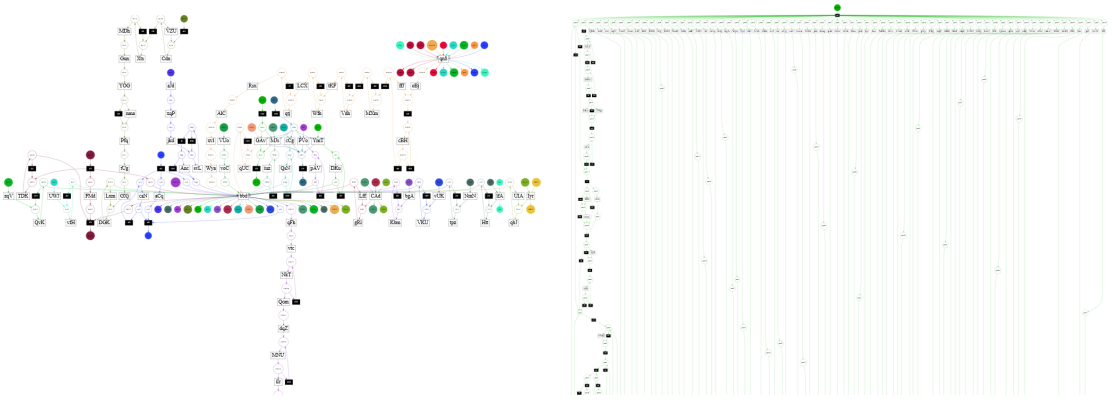

Figure 8: Discovered process model of a production process using the object-centric (left) and flattened event log (right). For the model discovered from the OCEL, different components follow their individual paths and are assembled into a merged component. For the model discovered from the flattened log, the process is mostly a flower model and the screenshot only shows part of the width of this flower model.

rediscovered: 100 partly concurrent activities can generate a large number of possible activity sequences, such that the data requirements of traditional process discovery would be beyond any feasible amount of produced products, i.e., the available sample size.

7. Conclusion

In this paper, we investigated the data requirement reduction that object-centric process discovery yields over traditional discovery. We categorize different real-life processes according to control-flow complexity across and within subprocesses and formally define these dimensions. We define an experimental framework that generates models across these dimensions and assess the reduction in data requirements when using object-centric discovery over traditional discovery. We show that the data requirements are drastically reduced, especially for low inter-object complexity process models like production processes or supply chains. To address limitations of the experimental framework w.r.t. the size of generated models we complement our evaluation with a large-scale production process case study. We show that object-centric discovery captures the production process much better than traditional process discovery. In this case, the data requirements of traditional discovery would even exceed the number of produced items, rendering traditional process discovery infeasible. Our results have significant implications for the application of process discovery to large-scale processes: Areas once infeasible for discovery due to large data requirements are now accessible using object-centric process mining.

Acknowledgments

We thank the Alexander von Humboldt (AvH) Stiftung for supporting our research. Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germanys Excellence Strategy—EXC-2023 Internet of Production390621612.

References

[1] M. Dumas, M. L. Rosa, J. Mendling, H. A. Reijers, Fundamentals of Business Process Management, Second Edition, Springer, 2018. doi:10.1007/978-3-662-56509-4.

[2] W. M. P. van der Aalst, Process Mining - Data Science in Action, Second Edition, Springer, 2016. doi:10.1007/978-3-662-49851-4.

[3] A. Augusto, R. Conforti, M. Dumas, M. L. Rosa, F. M. Maggi, A. Marrella, M. Mecella, A. Soo, Automated discovery of process models from event logs: Review and benchmark, IEEE Trans. Knowl. Data Eng. 31 (2019) 686–705. doi:10.1109/TKDE.2018.2841877.

[4] W. M. P. van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, A. J. M. M. Weijters, Workflow mining: A survey of issues and approaches, Data Knowl. Eng. 47 (2003) 237–267. doi:10.1016/S0169-023X(03)00066-1.

[5] J. D. Weerdt, M. T. Wynn, Foundations of process event data, in: Process Mining Handbook, Springer, 2022, pp. 193–211. doi:10.1007/978-3-031-08848-3_6.

[6] Y. Kim, J. Lee, Manufacturing strategy and production systems: An integrated framework, Journal of Operations Management 11 (1993) 3–15. doi:https://doi.org/10.1016/0272-6963(93)90029-O.

[7] G. Da Silveira, D. Borenstein, F. S. Fogliatto, Mass customization: Literature review and research directions, International Journal of Production Economics 72 (2001) 1–13. doi:https://doi.org/10.1016/S0925-5273(00)00079-7.

[8] S. Zeng, P. Melville, C. A. Lang, I. M. Boier-Martin, C. Murphy, Using predictive analysis to improve invoice-to-cash collection, in: SIGKDD, ACM, 2008, pp. 1043–1050. doi:10.1145/1401890.1402014.

[9] G. Schuh, A. Gützlaff, S. Cremer, S. Schmitz, A. Ayati, A data model to apply process mining in end-to-end order processing processes of manufacturing companies, in: IEEM, IEEE, 2020, pp. 151–155. doi:10.1109/IEEM45057.2020.9309946.

[10] M. Milgate, Supply chain complexity and delivery performance: an international exploratory study, Supply chain management: An international Journal 6 (2001) 106–118.

[11] W. M. P. van der Aalst, Object-centric process mining: Dealing with divergence and convergence in event data, in: SEFM, Springer, 2019, pp. 3–25. doi:10.1007/978-3-030-30446-1_1.

[12] J. N. Adams, D. Schuster, S. Schmitz, G. Schuh, W. M. P. van der Aalst, Defining cases and variants for object-centric event data, in: (ICPM), 2022, pp. 128–135. doi:10.1109/ICPM57379.2022.9980730.

[13] W. M. P. van der Aalst, A. Berti, Discovering object-centric Petri nets, Fundam. Informaticae 175 (2020) 1–40. doi:10.3233/FI-2020-1946.

[14] D. Cohn, R. Hull, Business artifacts: A data-centric approach to modeling business operations and processes, IEEE Data Eng. Bull. 32 (2009) 3–9.

[15] W. M. P. van der Aalst, A. Artale, M. Montali, S. Tritini, Object-centric behavioral constraints: Integrating data and declarative process modelling, in: Description Logics, CEUR-WS.org, 2017.

[16] D. Fahland, Describing behavior of processes with many-to-many interactions, in:

PETRI NETS, Springer, 2019, pp. 3–24. doi:10.1007/978-3-030-21571-2_1.
[17] M. Montali, A. Rivkin, Db-nets: On the marriage of colored Petri nets and relational databases, Trans. Petri Nets Other Model. Concurr. 12 (2017) 91–118. doi:10.1007/978-3-662-55862-1_5.
[18] S. Ghilardi, A. Gianola, M. Montali, A. Rivkin, Petri nets with parameterised data - modelling and verification, in: Business Process Management - 18th International Conference, BPM 2020, Seville, Spain, September 13-18, 2020, Proceedings, volume 12168 of Lecture Notes in Computer Science, Springer, 2020, pp. 55–74. doi:10.1007/978-3-030-58666-9_4.
[19] J. M. E. M. van der Werf, A. Rivkin, A. Polyvyanyy, M. Montali, Data and process resonance - identifier soundness for models of information systems, in: PETRI NETS, Springer, 2022, pp. 369–392. doi:10.1007/978-3-031-06653-5_19.
[20] E. H. J. Nooijen, B. F. van Dongen, D. Fahland, Automatic discovery of data-centric and artifact-centric processes, in: BPM Workshops, Springer, 2012, pp. 316–327. doi:10.1007/978-3-642-36285-9_36.
[21] L. Moctar-M'Baba, N. Assy, M. Sellami, W. Gaaloul, M. F. Nanne, Extracting artifact-centric event logs from blockchain applications, in: SCC, IEEE, 2022, pp. 274–283. doi:10.1109/SCC55611.2022.00048.
[22] G. Li, E. G. L. de Murillas, R. M. de Carvalho, W. M. P. van der Aalst, Extracting object-centric event logs to support process mining on databases, in: CAiSE Forum, Springer, 2018, pp. 182–199. doi:10.1007/978-3-319-92901-9_16.
[23] S. Esser, D. Fahland, Multi-dimensional event data in graph databases, J. Data Semant. 10 (2021) 109–141. doi:10.1007/s13740-021-00122-1.
[24] G. Li, R. M. de Carvalho, W. M. P. van der Aalst, Automatic discovery of object-centric behavioral constraint models, in: BIS, Springer, 2017, pp. 43–58. doi:10.1007/978-3-319-59336-4_4.
[25] J. N. Adams, W. M. P. van der Aalst, Precision and fitness in object-centric process mining, in: ICPM, IEEE, 2021, pp. 128–135. doi:10.1109/ICPM53251.2021.9576886.
[26] G. Park, J. N. Adams, W. M. P. van der Aalst, Opera: Object-centric performance analysis, in: ER, volume 13607, Springer, 2022, pp. 281–292. doi:10.1007/978-3-031-17995-2_20.
[27] A. Augusto, R. Conforti, M. Dumas, M. L. Rosa, A. Polyvyanyy, Split miner: automated discovery of accurate and simple business process models from event logs, Knowl. Inf. Syst. 59 (2019) 251–284. doi:10.1007/s10115-018-1214-x.
[28] S. J. J. Leemans, D. Fahland, W. M. P. van der Aalst, Discovering block-structured process models from event logs - A constructive approach, in: PETRI NETS, Springer, 2013, pp. 311–329. doi:10.1007/978-3-642-38697-8_17.
[29] A. Polyvyanyy, A. Moffat, L. García-Bañuelos, Bootstrapping generalization of process models discovered from event data, in: CAiSE, Springer, 2022, pp. 36–54. doi:10.1007/978-3-031-07472-1_3.
[30] T. Brockhoff, M. S. Uysal, I. Terrier, H. Göhner, W. M. P. van der Aalst, Analyzing multi-level bom-structured event data, in: ICPM Workshops, Springer, 2021, pp. 47–59. doi:10.1007/978-3-030-98581-3_4.