Journal of Intelligent Information Systems (2026) 64:941–964
https://doi.org/10.1007/s10844-025-01010-x

RESEARCH

Framework for grouping local process models

Viki Peeva[1] · **Wil M. P. van der Aalst**[1]

Received: 22 March 2024 / Revised: 7 November 2025 / Accepted: 12 November 2025 /
Published online: 13 January 2026

Abstract
Local Process Models (LPMs) are an underexplored concept in process mining. LPMs describe patterns in event data considering sequence, choice, concurrency, and loop. In recent years, process mining has proved successful in the analysis and improvement of operational processes. More often than not, surprising findings are found when one does not consider the full process, making LPMs and their discovery highly valuable. However, similar to other pattern mining approaches, LPM discovery algorithms face the problems of model explosion and model repetition, i.e., the algorithms may create hundreds if not thousands of LPMs, and subsets of them are close in structure or behavior. Practically, no analyst would be able to comb through thousands of LPMs leading to using a sample of LPMs that are easily accessible. The current sentiment is that the top-scoring LPMs form the optimal sample to be presented. However, different applications should demand a different optimal sample. With this work, we show that if the goal of the mined LPMs is to understand a process, using the top-scoring LPMs as an optimal sample is a poor choice because of high repetition. We propose a framework for grouping LPMs and creating an optimal sample by taking one representative LPM for each group. We measure similarity between models via established process model similarity measures or by comparing the context in which an LPM appears. The context is formed using data attributes available in the underlying event logs. We demonstrate the usefulness of grouping on multiple event logs by comparing repetition and coverage between samples comprised of the top-scoring models and the representatives of discovered groups.

Keywords Local process models · Process mining · Clustering · Data attributes

✉ Viki Peeva
peeva@pads.rwth-aachen.de

Wil M. P. van der Aalst
wvdaalst@pads.rwth-aachen.de

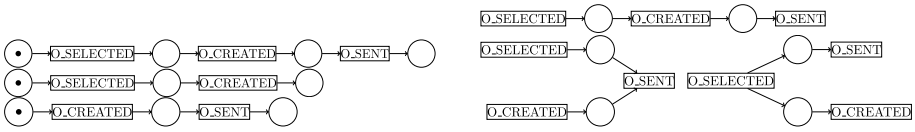
[1] Chair of Process and Data Science (PADS), RWTH Aachen University, Ahornstrasse 55, 52074 Aachen, North Rhine-Westphalia, Germany

1 Introduction

Process mining is a scientific discipline for discovering, monitoring, and improving processes via readily-available data from different data management systems. The three main pillars of process mining are process discovery, conformance checking, and process enhancement (van der Aalst, 2016). As the interest in process mining grows, the spectrum of processes to be analyzed expands. This triggers the need for novel techniques that provide insights into highly unstructured processes. Especially difficult, when analyzing such processes, is discovering a single well-defined model. Traditional process discovery techniques by trying to model the full range of variability would result in a so-called flower models (models that allow any behavior) or spaghetti models (models that because of modeling all the particularities in the process are difficult to read) (Tax et al., 2016b, See Fig. 7). To resolve this problem, one usually focuses on frequent behavior (van der Aalst, 2020). However, in some domains like healthcare and IoT (Munoz-Gama et al., 2022; Brzychczy et al., 2025), this is not possible, and better solutions are needed.

A relatively new field is Local Process Model (LPM) discovery (Tax et al., 2016b), where the idea is to build smaller process models explaining fragments of the behavior instead of one overall model. Yet, current LPM discovery approaches return hundreds or thousands of models (i.e., LPMs) for one event log (*model explosion*), with highly similar models repeating between them (*model repetition*) as shown in Fig. 1. Ideally, depending on the application in focus, one would consider a sample of the mined LPMs, which we refer to as *application-optimal* or *optimal* if the application is clear or irrelevant. Current LPM discovery approaches have introduced evaluation measures and rank the LPMs based on these. With this, they implicitly establish the optimal sample to be the top-scoring LPMs. However, here we argue that a better optimal sample can be chosen for the task of capturing the underlined process.

In this work, we propose grouping the discovered LPMs and for each group choosing one representative LPM. The set of representative LPMs would then characterize the optimal sample. Considering different needs and interests, the framework allows for two different perspectives when grouping the LPMs. On one hand, we allow for grouping LPMs based on the control flow by using already established process model similarity measures. On the other hand, we group LPMs based on the data attributes collected during the process execution. Meaning, LPMs are considered similar when they cover events appearing in a similar context. In the evaluation, we focus on comparing the top-scoring samples to the group representative samples. We compute diversity between the models in a sample and the count of events a sample covers as measurable way to quantify how well a set of LPMs capture the process. Additionally, we demonstrate the benefit of clustering LPMs by inspecting a smaller set of models on the real-life event logs *BPIC2012-res10939* and *Sepsis*. This paper



(a) Top-scoring LPMs discovered by [2]. (b) Top-scoring LPMs discovered by [6].

Fig. 1 Top-scoring LPMs discovered using Tax et al. (2016b) and Peeva et al. (2022) on the filtered and transformed *BPIC2012* event log for resource 10939 as explained in Tax et al. (2016b)

extends the work presented in Peeva and van der Aalst (2023) by incorporating the data perspective when grouping LPMs. Furthermore, we introduce event coverage on set level to assess how much the selected sample fits the event log. Finally, we extend the experiments to incorporate additional clustering parameters, and new similarity notions.

The reminder of the paper is structured as follows. In Section 2, we describe other grouping methods, and we cover related work of sub-parts of our framework. With Section 3, we establish the foundations necessary to follow the rest of the paper. In Section 4, we describe all contributions in detail, and we present the obtained results in Section 5. Finally, we discuss the limitations of the approach and future work in Section 6 and conclude the paper with Section 7.

2 Related work

Our framework for grouping similar LPMs together is based on multiple key aspects. Here, we consider related work for those individually. First, we consider LPM discovery approaches and how they incorporate data attributes in Section 2.1. Then, we discuss existing process model similarity measures and other frameworks for grouping LPMs or process models in Section 2.2.

2.1 Local process models

LPM discovery is positioned in-between traditional process discovery and episode and sequence mining (Mannila et al., 1997; Leemans & van der Aalst, 2014; Agrawal & Srikant, 1995; Srikant & Agrawal, 1996). In contrast to process discovery (Bergenthum et al., 2007; Carmona et al., 2008; Leemans et al., 2013; Mannel & van der Aalst, 2019; Wen et al., 2007; van der Werf et al., 2009; van Zelst et al., 2018), whose task is to discover one model that explains the traces of an event log from start to end, LPM discovery tries to mine a set of models each matching some particular behavioral patterns represented by subsequences in the event log. While episode and sequence mining can also be used to find frequent patterns in event logs, LPMs allow for more complex constructs like choices and loops. They were first introduced in Tax et al. (2016b) as a replacement for process discovery of highly unstructured processes. However, afterward, the use cases of LPMs expanded (see Deeva and Weerdt 2018; Delcoucq et al. 2020; Guzzo et al. 2022; Kirchner and Markovic 2018; Leemans et al. 2018; Mannhardt et al. 2018; Mannhardt and Tax 2017; Pijnenborg et al. 2021) and multiple approaches and extensions for LPM discovery followed (Dalmas et al., 2018; Acheli et al., 2019, 2022; Peeva et al., 2022; Peeva & van der Aalst, 2023; Brunings et al., 2022).

The initial approach by Tax et al. recursively extends process trees by appending more activities using different operators. In each iteration, because of monotonicity properties, LPMs are filtered by frequency, and only those satisfying a threshold proceed to the next phase to be extended with additional activities. The algorithm stops when there are no more frequent LPMs to be extended or new activities to be added. In Acheli et al. (2019) two existing LPMs (represented as process trees) differing only in one node are joined together to create a larger LPM. In the new model, the differing nodes are replaced with a process tree operator and are added to it as children. Similarly, using monotonicity properties,

at each step candidate LPMs are pruned out. The approach in Brunings et al. (2022) first finds frequently occurring gaped sequences and then discovers models on them. Finally, in Peeva et al. (2022), LPMs are created by joining single place nets into larger models. All approaches can represent the same or highly similar behavior with multiple, nearly identical, models. Therefore, focusing on frequent behavior results in clusters of repetitive models among the highly-ranked models.

All the approaches above (Acheli et al., 2019; Peeva et al., 2022; Tax et al., 2016b), except for (Brunings et al., 2022), suffer from the *model explosion* and *repetition* problems. In Brunings et al. (2022), the explosion problem is solved because for each frequent gapped sequences only one model is mined. However, this restricts the set of application scenarios the approach can be applied to (e.g., not suitable for event abstraction, trace classification and clustering, etc.), and even in the fewer built models, the repetition problem is still clearly visible. Moreover, for the most of the approaches there exist different extensions. For example, in Tax et al. (2016a) the authors try to decrease the complexity of Tax et al. (2016b) by using heuristics to project the event log on subset of activities before mining local process models. Both Tax et al. (2018) and Acheli et al. (2022) use data attributes to restrict the context for which LPMs are mined. Additionally, in Tax et al. (2018), it is possible to define utility functions on top of data attributes. One can then mine LPMs given a specific interest, by giving preference, i.e., higher rank, to LPMs that score well on the utility functions.

2.2 Grouping process models

This paper tries to decrease the repetition of similar LPMs by grouping them together. However, the challenge of organizing process models in model repositories (Yan et al., 2012), and querying (Jin et al., 2013; Awad et al., 2018), grouping (Ordóñez et al., 2017; Aiolli et al., 2011), or merging (Rosa et al., 2013; Sun et al., 2006) similar process models is not new. For example, Qiao et al. (2011); Sarno et al. (2013) group process models by performing graph-based partitioning such that two process models are connected in the graph if the computed similarity between the two process models exceeds a predefined threshold. Ordóñez et al. (2017) also presents an incremental clustering based on graph theory, but they do not prefilter connections between process models with low similarity, and the precomputed similarity matrix is directly used by the algorithm. In Jung and Bae (2006) process models are grouped using a hierarchical clustering algorithm, by converting them to activity transition vectors, and using the cosine measure to compute similarity. A hierarchical clustering method is also proposed in Ahn and Chang (2019), but the way similarity between processes is computed is specific for manufacturing processes. In Jung and Bae (2006); Bergmann et al. (2013) clustering is investigated for workflows, and Heinze et al. (2021); Ekanayake et al. (2012) consider Single-Entry Single-Exit (SESE) fragments in process models and try to discover approximate clones of such fragments in a process model repository.

Most of the grouping approaches rely on some kind of a process model similarity measure. Along with grouping, operations like retrieval, model querying, and model addition without creating duplicates are challenging and need a way to measure process model similarity. Therefore, various similarity measures are proposed in the literature and one can categorize them in different ways. Over the years there have been multitude of survey papers (Dijkman et al., 2013; Dumas et al., 2009; Schoknecht et al., 2017; Thaler et al., 2016;

Becker & Laue, 2012) covering this topic. Although there can be small differences in how survey papers categorize different similarity measures, all of them agree, the basic split is into measures that compare the structure of the process model, those that compare the behavior, or combine both. Subsequently, one can consider the level of abstraction used, e.g., complete language versus weak order relations. For more information, we direct the reader to the aforementioned survey papers. An alternative are similarity measures based on attributes. In Dijkman et al. (2011) a similarity measure that uses node attributes in a process model is proposed. However, this is different from our work, since we use data attributes available in the recorded behavior of the process. Such data attributes are used for clustering traces in van Zelst and Cao (2020), but we differentiate by using it to cluster models representing patterns in the underlying event log.

When it comes to LPMs, and the previously introduced LPM discovery approaches, the implementations of Tax et al. (2016b) and Peeva et al. (2022) offer grouping of the models by considering the common transition labels. Additionally, in Peeva and van der Aalst (2023) LPMs are grouped by using process model similarity measures, but to the best of our knowledge, data attributes have not yet been used to group process models or LPMs together.

3 Preliminaries

In this section, we provide essential background information needed to follow the rest of the paper.

3.1 Basic definitions

We define sets ($X = \{a, b\}$), multisets ($M = [a^2, b^3]$), sequences ($\sigma = \langle a, b, c \rangle$), and tuples ($t = (a, b, c)$) as usual. Given a set X, $\mathcal{P}(X)$ is the power set of X, X^* represents the set of all sequences over X, and $\mathbb{M}(X)$ is the set of all multisets over X. We use $\sigma(i)$ to denote the i-th element of the sequence σ, and $M(a) = 2$ to denote that item a appears twice in the multiset M.

Given a sequence $\sigma = \langle \sigma_1, \sigma_2, \ldots \sigma_n \rangle$, we extend σ with an additional element σ_{n+1} by writing $\sigma \cdot \sigma_{n+1}$. We call the sequence σ' a *subsequence* of σ, if and only if $\sigma' = \langle \sigma_l, \sigma_{l+1}, \ldots \sigma_m \rangle$ and $1 \leq l \leq m \leq n$ (we write $\sigma' \sqsubseteq \sigma$). We call σ' a *relaxed subsequence* (we write $\sigma' \underset{\sim}{\sqsubseteq} \sigma$) if and only if $\sigma' = \langle \sigma_{i_1}, \sigma_{i_2}, \ldots \sigma_{i_k} \rangle$ for $1 \leq i_1 < i_2 < \cdots < i_k \leq n$ and $k \geq 1$, i.e., we drop any number of elements from σ and keep the order for the rest. Lastly, given a sequence σ and a relaxed subsequence $\sigma' \underset{\sim}{\sqsubseteq} \sigma$, we call $s = \overline{\sigma'}^{\sigma}$ a *minimal over-sequence* of σ' in σ if and only if $s \sqsubseteq \sigma \wedge \sigma' \underset{\sim}{\sqsubseteq} s \wedge \nexists_{s'} (s' \sqsubseteq \sigma \wedge \sigma' \underset{\sim}{\sqsubseteq} s' \wedge |s'| < |s|)$.

To recalculate sets or multisets from other sets, multisets, or sequences, we use the $\{\cdot\}$ and $[\cdot]$ operators. We write $f : X \to Y$ to denote functions, and $dom(f) = X$, $rng(f) = Y$ to denote the domain and the range of the function f respectively. We use $f(X) = \{f(x) \mid x \in X\}$ (and $f(\sigma) = \langle f(\sigma(1)), f(\sigma(2)), \ldots, f(\sigma(n)) \rangle$) to apply the function f to every element in the set X (the sequence σ). Finally, we write $\sigma_{\restriction X}$ to denote the projection of the sequence σ on the set X.

3.2 Process mining

Normally, the collected data used for process analysis is transformed into *event logs*. In traditional process mining, event logs focus on a single case notion where each event occurs exactly for one case. Events are denoted by the executed activity, the timestamp when the event occurred, and for which case they were executed. Additionally, event logs can contain further data attributes, either on case or event level.

In Definition 1, we formally define *event logs*, and with Definition 2 we extract the set of all traces of the event log, where a trace is defined as a sequence of events.

Definition 1 (Event Log) Let $\mathcal{U}_{ev}$ be the universe of events, $\mathcal{U}_{attr}$ the universe of attribute names, and $\mathcal{U}_{val}$ the universe of attribute values. An *event log* $L = (E, Attr, \pi)$ is a tuple, with $E \subseteq \mathcal{U}_{ev}$ the set of events, $Attr$ a set of attribute names such that $\{act, case, time\} \subseteq Attr$, and $\pi : E \times Attr \not\to \mathcal{U}_{val}$ a (partial) function assigning to an event $e \in E$ and attribute name $attr \in Attr$ pair, a value $\pi(e, attr) \in \mathcal{U}_{val}$.

Definition 2 (Traces) Given an event log $L = (E, Attr, \pi)$ and a case $c \in \pi(E, case)$,[1] we define $E_c^L = \{e \in E \mid \pi(e, case) = c\}$ as the set of all events of the case c. Now, the *trace* of c is a sequence of its events ordered by time, i.e., $\rho_c^L = \langle e_1^c, e_2^c, ..., e_{|E_c^L|}^c \rangle$ where $\forall_{1 \leq i < j \leq |E_c^L|} \pi(e_i^c, time) \leq \pi(e_j^c, time)$. Moreover, we write $traces(L) = \{\rho_c^L \mid c \in \pi(E, case)\}$ to denote all traces for the event log L.

Let us consider the toy event log $L_{gp} = (E, Attr, \pi)$ simulating a Real Time Strategy (RTS) game in Table 1. In the event log, one case is considered one gameplay from the perspective of one user, meaning the attribute *Game Id* represents the case. The *Activity* and *Timestamp* are the main attributes available for events, *Player* and *Experience Tier* are additional attributes on case level, and *Idle Units* is an attribute on event level. Per Definition 1, $\pi(e6, case) = 202$, the $\pi(e6, player) = Trolldor$, and $\pi(e6, \text{idle-units}) = 16$. The set of all events for the case 202 is $E_{202}^{L_{gp}} = \{e1, e2, e4, e5, e6\} \subseteq E$, and the traces of the event log, we obtain with $traces(L_{gp}) = \{\langle e1, e2, e4, e5, e6 \rangle, \langle e3, e7 \rangle\}$.

The behavior recorded in logs can be modeled using different notations, such as DFG, process trees, BPMN, Petri nets, etc. In this work, we focus on Petri nets, and more precisely on labeled Petri nets which we define in Definition 3. Note that a transition $t \in T$ with

Table 1 A toy event log L_{gp} for game play. The *Event Id* column represents the events, and the rest of the columns represent different event attributes

Event Id	Game Id (*case*)	Activity (*act*)	Timestamp (*time*)	Player (*player*)	Experience Tier (*tier*)	Idle Units (*idle-units*)
*e*1	202	**S**elect	2024-02-13 16:57:21.030	Trolldor	Medium	27
*e*2	202	**M**ove	2024-02-13 16:57:23.047	Trolldor	Medium	23
*e*3	243	**M**ove	2024-02-13 16:57:24.025	DustGod	Advanced	1
*e*4	202	**S**elect	2024-02-13 16:57:28.015	Trolldor	Medium	31
*e*5	202	**C**ollect	2024-02-13 16:57:29.001	Trolldor	Medium	16
*e*6	202	**A**ttack	2024-02-13 16:58:03.066	Trolldor	Medium	16
*e*7	243	**S**elect	2024-02-13 16:58:04.011	DustGod	Advanced	6

[1] We overload the function π on sets of events as explained in Section 3.1.

$l(t) = \tau$ is called silent, and if two transitions share a common label, i.e., $t_1, t_2 \in T$ such that $t_1 \neq t_2$ and $l(t_1) = l(t_2)$, we call them duplicate transitions.

Definition 3 (Labeled Petri net) Given the universe of activities, $\mathcal{A}$, a *labeled Petri net* $N = (P, T, F, l)$ is a tuple, where P is a set of places and T is a set of transitions such that $P \cap T = \emptyset$. $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation, and $l : T \to \mathcal{A} \cup \{\tau\}$ the labeling function.

Now, given a node $x \in P \cup T$, we define the *preset* of x as $\bullet x = \{y \in P \cup T \mid (y, x) \in F\}$ and the *postset* of x as $x\bullet = \{y \in P \cup T \mid (x, y) \in F\}$.

To attach behavior to labeled Petri nets we use a *marking* and the *firing rule*. A marking M denotes the state of a Petri net $N = (P, T, F, l)$ as a multiset of places ($M \in \mathbb{M}(P)$) and the firing rule allows for changes between states (i.e., markings). Given a marking M, a transition t is *enabled* in the marking M if and only if $\bullet t \subseteq M$. If a transition t is enabled in the marking M, it can fire and change the state of the net to a new marking $M' = (M \setminus \bullet t) \cup t\bullet$. We write $M \xrightarrow{t} M'$. A sequence of transitions $\sigma = \langle t_1, \ldots, t_n \rangle \in T^*$ is enabled in M and by firing it marking M' is reached if and only if there exist M_0, M_1, , M_n such that $M_0 = M$, $M_n = M'$, and $M_{i-1} \xrightarrow{t_i} M_i$ for $1 \leq i \leq n$. We write $M \xrightarrow{\sigma} M'$.

Now, we formally define an *accepting Petri net* in Definition 4 and the set of all its firing sequences in Definition 5.

Definition 4 (Accepting Labeled Petri Net) An *accepting labeled Petri net* is a triple (N, M_i, M_f) such that $N = (P, T, F, l)$ is a labeled Petri net, $M_i \in \mathbb{M}(P)$ is the initial marking, and $M_f \in \mathbb{M}(P)$ is the final marking.

Definition 5 (Complete Firing Sequences) Let $\mathrm{AN} = (N, M_i, M_f)$ be an accepting Petri net, $\mathcal{F}(\mathrm{AN}) = \{\sigma \in T^* \mid M_i \xrightarrow{\sigma} M_f\}$ is its set of *complete firing sequences*.

The *language* of an accepting Petri net $\mathrm{AN} = (N, M_i, M_f)$ is obtained by projecting all complete firing sequences on the transition labels and removing τ-skips, i.e., $\mathcal{L}(\mathrm{AN}) = \{l(\sigma)_{\restriction_{\mathcal{A}}} \mid \sigma \in \mathcal{F}(\mathrm{AN})\}$. The sequences in $\mathcal{L}(\mathrm{AN})$ we also call *traces* of the net. We use $\mathcal{L}^n(\mathrm{AN}) = \{l(\sigma)_{\restriction_{\mathcal{A}}} \mid \sigma \in \mathcal{F}(\mathrm{AN}) \wedge |\sigma| \leq n\}$ to denote the language restricting to complete firing sequences of length at most n.

3.3 Local process models

We now focus on formalizing LPMs. Similar to end-to-end process models, LPMs can be represented by any modeling language (Petri nets, process trees, BPMNs, etc.), however, in this work we restrict to a subclass of accepting labeled Petri nets.

We show an example LPM in Fig. 2, and a formal definition in Definition 6.

Definition 6 (Local Process Models) A *Local Process Model (LPM)* is an accepting labeled Petri net $\mathrm{lpm} = (N_{\mathrm{lpm}}, M_i, M_f)$ such that $N_{\mathrm{lpm}} = (P, T, F, l)$ is a labeled Petri net that satisfies the following restrictions:

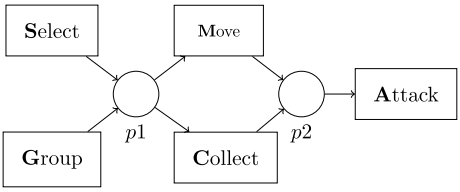


Fig. 2 Example LPM lpm_{gp} for the Game Play event log in Table 1

1. $\forall_{x,x' \in P \cup T}\, \exists_{\langle x_1,\ldots,x_n \rangle} (x = x_1 \wedge x' = x_n \wedge \forall_{1 \leq i < n}((x_i, x_{i+1}) \in F) \vee (x_{i+1}, x_i) \in F))$, i.e., there is only one connected component, and
2. $\forall_{p \in P}(\bullet p \neq \emptyset \wedge p\bullet \neq \emptyset)$, i.e., each place has at least one incoming and outgoing arc,

and the initial and final marking are empty, i.e., $M_i = [\,]$ and $M_f = [\,]$. We use $\mathcal{U}_{\mathrm{LPM}}$ to denote the universe of such *LPMs*.

We use $T_{in}(N_{\mathrm{lpm}}) = \{t \in T_{\mathrm{lpm}} \mid \bullet t = \emptyset\}$to denote the transitions that have an empty preset, and we call them *unrestricted transitions*. In the case of the LPM $\mathrm{lpm}_{gp} = (N_{\mathrm{lpm}_{gp}}, [\,], [\,])$, $P_{\mathrm{lpm}_{gp}} = \{p1, p2\}$, $T_{\mathrm{lpm}_{gp}} = \{S, G, M, C, A\}$,$F_{\mathrm{lpm}_{gp}}\{(S, p1), (G, p1), (p1, M), (p1, C), (M, p2), (C, p2), (p2, A)\}$, and $T_{in}(\mathrm{lpm}_{gp}) = \{S, G\}$.

We define the set of complete valid firing sequences of such LPMs in Definition 7 by restricting that each place in the net can receive at most one token from unrestricted transitions. Meaning, for the LPM from before, we have $\mathcal{F}_{\mathrm{LPM}}(\mathrm{lpm}_{gp}) = \{\langle S, M, A\rangle, \langle G, M, A\rangle, \langle S, C, A\rangle, \langle S, M, A\rangle\}$. Note, the firing sequence $\langle S, M, G, C, A\rangle$ is not a valid firing sequence, because two unrestricted transitions, namely S and M, put a token in $p1$.

Definition 7 (LPM Behavior) Given an LPM $\mathrm{lpm} = (N_{\mathrm{lpm}}, M_i, M_f)$ such that $N_{\mathrm{lpm}} = (P, T, F, l)$, we define $\mathcal{F}_{\mathrm{LPM}}(\mathrm{lpm}) = \{\sigma \in \mathcal{F}(\mathrm{lpm}) \mid \forall_{1 \leq i < j \leq |\sigma|}(\sigma(i) \in T_{in} \wedge \sigma(j) \in T_{in} \implies \sigma(i) \bullet \cap \sigma(j)\bullet = \emptyset)\}$ as all valid complete firing sequences of lpm.

We obtain the *language* $\mathcal{L}(\mathrm{lpm})$ of an LPM lpm analogous to the language of accepting Petri nets, but now we only consider the valid complete firing sequences as per Definition 7. We use $\mathcal{L}^n(\mathrm{lpm})$ to denote the language restricting to valid complete firing sequences of length at most n.

Given an LPM lpm and an event log L, we can extract all occurrences of the LPM in the event log. To achieve this, we define a function $\lambda(\mathrm{lpm}, L)$ in Definition 8.

Definition 8 (LPM Occurrence List) Given an *LPM* lpm and an event log L, we define $\lambda(lpm, L) = \{s \mid \exists_{\rho \in traces(L)}(s \sqsubseteq \rho \wedge \pi(s, act) \in \mathcal{L}(lpm))\}$[2] to be the LPM *occurrence list* of lpm in L.

Additionally, we extend λ with λ_n such that the length of the minimal event over-sequence does not exceed n, i.e., $\lambda_n(lpm, L) = \{s \mid \exists_{\rho \in traces(L)}(s \sqsubseteq \rho \wedge |\overline{s}^{\rho}| \leq n \wedge \pi(s, act) \in \mathcal{L}(lpm))\}$

[2] We overload the function π on sequences of events as explained in Section 3.1.

Consider again the LPM in Fig. 2 and the event log extract given in Table 1. The LPM occurrence list for lpm_{gp} and L_{gp} is $\lambda(\mathrm{lpm}_{gp}, L_{gp}) = \{\langle e_1, e_2, e_6\rangle, \langle e_4, e_5, e_6\rangle\}$ and $\lambda_4(\mathrm{lpm}_{gp}, L_{gp}) = \{\langle e4, e5, e6\rangle\}$ since $|\overline{\langle e1, e2, e6\rangle}^{\rho_{202}^{L_{gp}}}| = |\langle e1, e2, e4, e5, e6\rangle| = 5 > 4$.

With the LPM occurrence list, we can measure conformance to an event log L and rank the LPMs. The ranking can also take different quality measures into consideration, such as fitness, precision, and simplicity. We write $\mathrm{rank} \in \mathcal{U}_{\mathrm{LPM}} \not\to \mathbb{N}$ to denote a ranking function, and $\mathcal{U}_{\mathrm{rank}}$ to denote the universe of all such ranking functions.

4 The LPM grouping framework

In this part, we detail the proposed approach of grouping LPMs, starting from a set of LPMs and the event log on which they were discovered, all the way until the final LPM groups and the chosen representatives for each group.

We give an overview of the grouping framework in Fig. 3. The framework requires a set of LPMs LPM and an event log L as input. In the end, it returns a partitioning of the original LPM set $\sqcap^{\mathrm{LPM}}$ and a set of representative LPMs $\sqcap^{\mathrm{LPM}}_{\mathrm{repr}}$, i.e., a subset of LPMs of the original set, each one belonging to exactly one group. We illustrate the entire approach by splitting it to individual steps. In the first step we compute the distance between two LPMs using process model similarity measures, data attributes, and a combination of both. Next, we formally define clustering the set of LPMs by using the precomputed distances. Finally, we choose one representative LPM for each cluster and create the set of LPM representatives.

4.1 Extraction of LPM pairwise distances

In this step, we start with a set of LPMs LPM and an event log L. For each LPM pair we compute a distance between 0 and 1 using a distance function $\mathrm{dist}_x \in \mathcal{U}_{\mathrm{dist}}$ (where $\mathcal{U}_{\mathrm{dist}} : \mathcal{U}_{\mathrm{LPM}} \times \mathcal{U}_{\mathrm{LPM}} \to [0, 1]$). The framework allows for three options for distance extraction: process model similarity measures, data attributes, and mixed. In the following subsection we explain the different distance extraction methods.

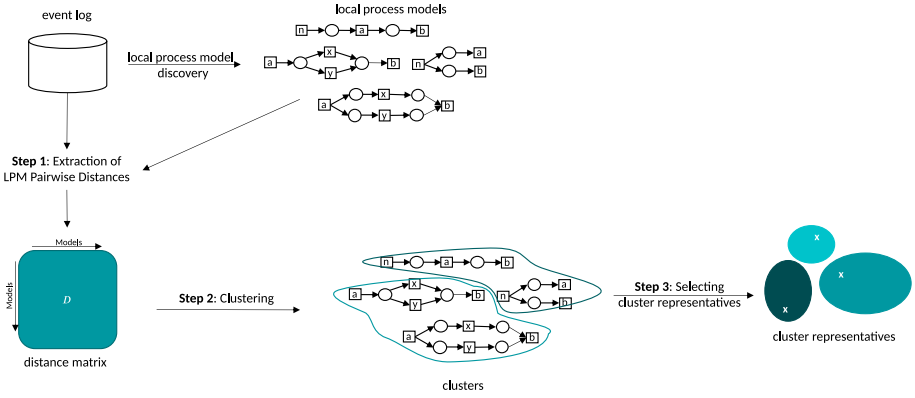


Fig. 3 Illustration of the proposed three-step framework

4.1.1 Process model similarity measures

As discussed in Section 2.2, the literature agrees on two orthogonal categorizations of process model similarity measures. On the one hand, we have measures that focus on the model's structure or the model's behavior, or we include both. On the other hand, there are different abstraction levels at which a measure can consider the models, e.g., complete language vs a directly-follow or an eventually-follow relationship abstraction. Here, we describe five measures that are representative of the abovementioned categories. We chose these five measures as a starting point for the grouping framework. However, the framework per design is not restrictive regarding the used process model similarity measures.

To introduce the similarity measures, we assume we are given two LPMs $\mathrm{lpm}_A = (N^A_{\mathrm{lpm}}, [\,], [\,])$ s.t. $N^A_{\mathrm{lpm}} = (P_A, T_A, F_A, l_A)$ and $\mathrm{lpm}_B = (N^B_{\mathrm{lpm}}, [\,], [\,])$ s.t. $N^B_{\mathrm{lpm}} = (P_B, T_B, F_B, l_B)$. In the following, we illustrate the measures we use in this work with the help of lpm_A and lpm_B.

Transition Label (TL) overlap is the simplest measure we investigate. It computes the similarity between models by quantifying the TL overlap as shown below.

$$\mathrm{sim}_{\mathrm{tl}}(\mathrm{lpm}_A, \mathrm{lpm}_B) = \frac{2 * |l_A(T_A) \cap l_B(T_B)|}{|l_A(T_A)| + |l_B(T_B)|}$$

Node Matching (NM) is somewhat more complex, in that it includes place overlap as well. We use this measure to represent structural measures using abstraction. The measure calculates the similarity between two models by combining TL overlap and place matching between the nets. We assign to each pair of places a matching gain $g(p_1, p_2) = \frac{1}{2} * \frac{2*|l_A(\bullet p_1) \cap l_B(\bullet p_2)|}{|l_A(\bullet p_1)| + |l_B(\bullet p_2)|} + \frac{1}{2} * \frac{2*|l_A(p_1 \bullet) \cap l_B(p_2 \bullet)|}{|l_A(p_1 \bullet)| + |l_B(p_2 \bullet)|}$, and we use the Hungarian algorithm (Kuhn, 2010) to solve the assignment problem. We use G_{places} to represent the gain of the optimal assignment. Then, we define the measure as

$$\mathrm{sim}_{\mathrm{nm}}(\mathrm{lpm}_A, \mathrm{lpm}_B) = \frac{2 * |l_A(T_A) \cap l_B(T_B)| + 2 * G_{\mathrm{places}}}{|l_A(T_A)| + |l_B(T_B)| + |P_A| + |P_B|}$$

Eventually-Follow (EF) overlap is a behavioral abstraction measure that measures the overlap of the eventually-follows relation in the languages of the two models. We calculate it as

$$\mathrm{sim}^n_{\mathrm{ef}}(\mathrm{lpm}_A, \mathrm{lpm}_B) = \frac{2 * |\mathrm{EF}^n_A \cap \mathrm{EF}^n_B|}{|\mathrm{EF}^n_A| + |\mathrm{EF}^n_B|}$$

such that $\mathrm{EF}^n_A = \{(a, b) \mid \exists_{\rho \in \mathcal{L}^n(\mathrm{lpm}_A)}(\exists_{1 \le i < j \le |\rho|}(a = \rho(i) \wedge b = \rho(j)))\}$ and EF^n_B is defined correspondingly.

Trace Matching (TM) is a similarity measure based on the behavior of the process models. Given the set of all traces of lpm_A and lpm_B restricted to a length at most *n*, i.e., $\mathcal{L}^n(\mathrm{lpm}_A)$ and $\mathcal{L}^n(\mathrm{lpm}_B)$, we assign each pair of traces $\rho_1 \in \mathcal{L}^n(\mathrm{lpm}_A)$ and $\rho_2 \in \mathcal{L}^n(\mathrm{lpm}_B)$ a matching gain $g(\rho_1, \rho_2)$. We compute the gain between two traces by inverting the normalized Levenshtein distance. Then, we use the Hungarian algorithm to solve the assignment prob-

lem. We use G_{traces} to represent the gain of the optimal trace assignment. At last, we define the measure as

$$sim^{n}_{\text{tm}}(\text{lpm}_A, \text{lpm}_B) = \frac{2 * G_{\text{traces}}}{|\mathcal{L}^n(\text{lpm}_A)| + |\mathcal{L}^n(\text{lpm}_B)|}$$

Finally, *Graph Edit Distance (GED)* represents advanced structural measures. It calculates model similarity by computing the GED between two model. In our work, we use the measure as defined in Dijkman et al. (2009, Definition 4).

Note that all similarity measures compute values between 0 and 1. In Definition 9 we formally define the conversion to a distance measure.

Definition 9 (LPMs Distance For Process Model Similarity Measures) Given two *LPMs* lpm_A and lpm_B, and a similarity measure sim, we compute the distance between them by inverting the computed similarity. Formally, $\text{dist}(\text{lpm}_A, \text{lpm}_B) = 1 - \text{sim}(\text{lpm}_A, \text{lpm}_B)$.

4.1.2 LPM data attributes measures

In this section, we use the events covered by an LPM and the available data attributes to represent the context in which an LPM appears in the process. To get the events covered by an LPM lpm, we use the occurrence list from Definition 8. In the continuation, we first describe encoding individual data attributes, followed by the process of appending the individual contexts to form the collective one. Then, we define how to compute the distance between two LPMs by computing the distance between their collective context vectors.

Context encoding of data attributes Given an event log $L = (E, Attr, \pi)$, an LPM lpm, and a data attribute $attr \in Attr$, we obtain the multiset of events covered by lpm in L using the occurrence list function, that is $E^L_{\text{lpm}} = [e \in s \mid s \in \lambda(\text{lpm}, L)]$ are the *covered events*. Then, we compute the *value multiset* for $attr$ and lpm in L by projecting the covered events on the attribute of interest $attr$: $Val^{(L,\text{lpm})}_{attr} = \pi(E^L_{lpm},\ attr)$. Now, given the value multiset, we extract a data vector $\vec{d}^{(lpm,\ L)}_{attr}$ to represent the data attribute context for lpm in L. Depending on whether the data attribute is categorical or numerical we use different strategy to build the data vector, as explained below.

Numerical Data Attributes. Given the value multiset Val_{attr_n} (short V) for a numerical data attribute $attr_n$, we then extract four features to represent the data.

- The minimum value $\min(V) = \min_{x \in V} x$
- The maximum value $\max(V) = \max_{x \in V} x$
- The mean: $\text{mean}(V) = \frac{\sum_{x \in V} x}{|V|}$

- The median:[3] $\text{median}(V) = \begin{cases} \sigma_V(\frac{|V|-1}{2}), & \text{if } |V| \equiv 1 \mod 2 \\ \frac{\sigma_V(\frac{|V|}{2})+\sigma_V(\frac{|V|-2}{2})}{2}, & \text{if } |V| \equiv 0 \mod 2 \end{cases}$

From these four features, we create the data vector that represents the numerical data attribute in the still to be built collective context for the LPM. For example, consider the numerical attribute *Idle Units* in L_{gp} (Table 1). The computed multiset for this data attribute and the LPM lpm_{gp} in Fig. 2 is $Val_{idle-units}^{(\text{lpm}_{gp},L_{gp})} = \pi([e1^1, e2^1, e4^1, e5^1, e6^2], \text{idle-units}) = [27, 23, 31, 16^3]$. The resulting data vector consisting of the four features is $\left[16 \;\; 31 \;\; \frac{\sum_{x \in [27,23,31,16^3]} x}{6} \;\; \frac{16+23}{2}\right]$.

Categorical Data Attributes. Given the value multiset $\mathcal{U}_{valattr_c}$ for a categorical data attribute $attr_c$, we create a feature vector where each distinct value represents a feature and the values are the count of that value appearing in the multiset. Consider the categorical attribute *Experience Tier* in L_{gp} (Table 1). The computed multiset for this data attribute and the LPM lpm_{gp} in Fig. 2 is $Val_{tier}^{(\text{lpm}_{gp},L_{gp})} = \pi([e1^1, e2^1, e4^1, e5^1, e6^2], \text{tier}) = [medium^6]$. We then convert the computed multiset to the data vector $[0 \quad 6]$, where the first value corresponds to the data value *advanced* and the second value to the data value *medium*.

Encoding of the collective occurrence context In this part, we use the context encoding of single data attributes to produce the collective occurrence context by first normalizing and then joining the single context encodings. Let $D^{(\text{lpm},L)}$ be the set of data vectors of all data attributes $attr \in Attr$ for one LPM, and $D^L = \bigcup_{\text{lpm} \in \text{LPM}} D^{(\text{lpm},L)}$ the set of data vectors of all data attributes for all LPMs in LPM. To ensure that each data attribute contributes the same to the collective context, we normalize the computed data vectors $D^{(\text{lpm},L)}$ across all attributes. Since the data vectors for different data attributes vary in length, we normalize them by its length, i.e., $\vec{d} = \frac{\vec{d}}{\|\vec{d}\|}$ where $\vec{d} \in D^L$. This way, each data attribute plays an equal role in the collective context.

Finally, in Definition 10, we define how we build the collective context for one LPM, by first ordering the set of data vectors $D^{(\text{lpm},L)}$, so that each position always represents the same feature across all LPMs, and concatenating them together. We use $(D^{(\text{lpm},L)}, \leq)$ to denote the ordered set.

Definition 10 (Collective Context) Given an event log L, an *LPM* lpm, a set of data attributes $Attr$, the precomputed and normalized set of all data vectors $D^{(\text{lpm},L)}$ for those attributes, and an ordering $\leq_{Attr}$, we call the data vector $\vec{cc}^{(\text{lpm},L)} = \cdot(D^{(\text{lpm},L)}, \leq_{Attr})$, formed as concatenation of all the data vectors, the *collective context* of the *LPM* lpm in L.

Distance computation between data vectors To compute the distance between two LPMs, we employ distance functions that compute the distance between two data vectors representing their respective collective contexts. Well-known distance functions of this form are the Euclidean distance, Chebyshev distance, cosine distance, Minkowski distance, etc. We use the notation dist_v to represent such functions without explicitly selecting one. Most of these distances do not give a value between 0 and 1, so we additionally normalize them.

[3] To demonstrate the median computation, we use the notation σ_M to represent the sequence computed from a multiset $M \in \mathbb{M}(\mathbb{N})$ using the total order defined by the $\leq$ operator.

Finally, in Definition 11 we instantiate the distance computation between two LPMs in Definition 11 when we consider data attributes.

Definition 11 (LPMs Distance For Data Attributes) Given an event log L, and two LPMs lpm_A and lpm_B, we compute the distance between lpm_A and lpm_B as the distance between the data vectors of their corresponding collective contexts $\vec{cc}^{(\mathrm{lpm}_A,L)}$ and $\vec{cc}^{(\mathrm{lpm}_B,L)}$. Formally, $\mathrm{dist}(\mathrm{lpm}_A, \mathrm{lpm}_B) = \mathrm{dist}_v(\vec{cc}^{(\mathrm{lpm}_A,L)}, \vec{cc}^{(\mathrm{lpm}_B,L)})$, where dist_v computes the normalized distance between two data vectors.

4.1.3 Combined measures

To let both similarities in the process model and in the context of which LPMs appear weigh-in in building the groups, we allow for computing a mixed distance between two LPMs, lpm_A and lpm_B. To achieve this, we introduce a threshold $\tau \in [0, 1]$ that controls the influence of the distance computed using process model similarity measures $dist^{pms}$ versus the distance computed from the collective contexts $dist^{cc}$. We compute the mixed distance as $dist^{mx} = \tau * dist^{pms} + (1 - \tau) * dist^{cc}$.

We also extend the notation of mixed distance, by allowing multiple distances computed from process model similarity measures or data vectors for specific subsets of attributes. Let $\tau_i \in [0, 1]$ for $i \in \{1, ...k\}$ such that $\sum_{i=1}^{k} \tau_i = 1$. We define the mixed distance as $dist^{mx} = \sum_{i=1}^{k} \tau_i * dist_i$ where $dist_i \in \{dist_x^{pms}, dist_x^{cc}\}$ and x denotes the concrete process model similarity measure used or the subset of attributes to compute the data vector.

4.2 Clustering

In the clustering step, we use a distance function $dist \in \mathcal{U}_{\mathrm{dist}}$ from the ones defined previously, to partition the set LPM in groups. To formalize this, we define a *clustering algorithm* in Definition 12.

Definition 12 (Clustering Algorithm) Let $\mathcal{U}_{\sqcap} = \{X \subseteq \mathcal{P}(\mathcal{U}_{\mathrm{LPM}})\}$ be the universe of cluster sets. A function $\mathrm{clust} \in \mathcal{P}(\mathcal{U}_{\mathrm{LPM}}) \times \mathcal{U}_{\mathrm{dist}} \not\to \mathcal{U}_{\sqcap}$ is a *clustering algorithm* if and only if for any *LPM* set $\mathrm{LPM} \in \mathcal{P}(\mathcal{U}_{\mathrm{LPM}})$ and a distance function $\mathrm{dist} \in \mathcal{U}_{\mathrm{dist}}$, $X_{\mathrm{LPM}} = \mathrm{clust}(\mathrm{LPM}, \mathrm{dist})$ satisfies the following two properties:

- $\bigcup X_{\mathrm{LPM}} = \mathrm{LPM}$
- $\forall_{X_i, X_j \in X_{\mathrm{LPM}}} X_i \cap X_j = \emptyset$

To denote a clustering algorithm given some set of parameters P, we write clust_P.
The goal of the clustering algorithm is to return the LPMs in homogeneous groups, such that the similarity is high within the individual groups and low between them. One clustering algorithm can produce different cluster sets for the same distance function and set of LPMs based on the parameters P.

Now, given the predefined distance function dist, an LPM set LPM, and a *clustering algorithm* clust_P we compute a set of clusters $\sqcap^{(\mathrm{clust}_P, \mathrm{LPM}, \mathrm{dist})} = \mathrm{clust}_P(\mathrm{LPM}, \mathrm{dist}) = X_{\mathrm{LPM}}$ such that $X_{\mathrm{LPM}} \in \mathcal{U}_{\sqcap}$. We overload the notation $\sqcap^{(\mathrm{clust}_P, \mathrm{LPM}, \mathrm{dist})}(\mathrm{lpm})$ to denote

the cluster $X \in X_{\mathrm{LPM}}$ for which the LPM lpm belongs, i.e., $\mathrm{lpm} \in X$. That is, it holds $\mathrm{lpm} \in \sqcap^{(\mathrm{clust}_P,\mathrm{LPM},\mathrm{dist})}(\mathrm{lpm}) \in \sqcap^{(\mathrm{clust}_P,\mathrm{LPM},\mathrm{dist})}$.

4.3 Computing cluster representatives

In Step 3, we use the computed cluster set $\sqcap^{(\mathrm{clust}_P,\mathrm{LPM},\mathrm{dist})}$ from Step 2, to calculate the *representative set* $\sqcap_{\mathrm{repr}}^{(\mathrm{clust}_P,\mathrm{LPM},\mathrm{dist})} \subseteq \mathrm{LPM}$ in which we keep only one representative LPM per cluster. In Definition 13, we formally define *representative projection* as a function that maps a set of LPMs to one model.

Definition 13 (LPM Set Representative) Given a ranking function $\mathrm{rank} \in \mathcal{U}_{\mathrm{rank}}$ and a set of LPMs LPM, we define $\mathrm{repr}(\mathrm{LPM}, \mathrm{rank}) = \arg\max_{\mathrm{lpm} \in \mathrm{LPM}} \mathrm{rank}(\mathrm{lpm})$ to be the *representative* of LPM given rank.

Now, for the set of LPMs $\mathrm{LPM} \subseteq \mathcal{U}_{\mathrm{LPM}}$ and a ranking function rank, we create the set $\sqcap_{\mathrm{repr}}^{(\mathrm{clust}_P,\mathrm{LPM},\mathrm{dist})} = \{\mathrm{repr}(\mathrm{LPM}_i, \mathrm{rank}) \mid \mathrm{LPM}_i = \sqcap^{(\mathrm{clust}_P,\mathrm{LPM},\mathrm{dist})}(\mathrm{lpm}) \wedge \mathrm{lpm} \in \mathrm{LPM}\}$, and we call it the *cluster representatives*. This way, we significantly reduce the number of LPMs from the original set LPM, but still keep the essence of the entire set. As a ranking function we use $\mathrm{rank}_L^{\mathrm{freq}}(\mathrm{lpm}) = |\lambda(\mathrm{lpm}, L)|$ to take the most frequent model, or $\mathrm{rank}^{\mathrm{dist}}(\mathrm{lpm}) = 1 - \frac{\sum_{\mathrm{lpm}' \in \sqcap(\mathrm{lpm})} \mathrm{dist}(\mathrm{lpm},\mathrm{lpm}')}{|\sqcap(\mathrm{lpm})|}$ to take the medoid LPM, i.e., the model with the minimal distance to all other models in the cluster.

4.4 Conclusion

To summarize, in our grouping framework we start with an event log L and a set of LPMs LPM. The framework allows for computing the distance between LPMs using established process model similarity measures or by representing each LPM as a data vector summarizing data attributes of interest for the covered events. Given we chose one of the ways to compute the distance, we use a clustering algorithm to group similar LPMs together. Finally, given a ranking function rank, for each group we extract one LPM that serves as a representative of that group. The final outputs of the framework are the set of clusters and the cluster representatives.

5 Evaluation of the framework

To evaluate the proposed framework, we consider several perspectives. First, to assess the impact of the framework, we compare the top-scoring samples of LPMs with representative samples (for different sample sizes). Second, we evaluate the cohesiveness of the clustering results and if those correlate with the repetitiveness and coverage computed before. Third, we report the running time of the distance computation and clustering relative to the discovery approach. Finally, we examine three representative LPMs for the event log used as motivational example in Section 1.

We organize the rest of the section by first presenting implementation and the experimental setup. Then, we have a section for each of the perspectives we evaluate.

5.1 Implementation

We implemented the entire framework in `ProM`, as part of the `LocalProcessModel-DiscoveryByCombiningPlaces` package.[4] The same package also contains the LPM discovery approach presented in Peeva et al. (2022).

In the implementation, we chose hierarchical clustering algorithm for grouping the LPMs and considered number of clusters and linkage as possible parameters. Linkage determines how the distance between two clusters containing multiple models is calculated. For the hierarchical clustering algorithm and everything around it, we use the Smile library (Li, 2014). All other computations are directly implemented in the package itself.

5.2 Experimental setup

We perform all experiments on the event logs and LPM sets in Table 2. For computing distances, we use the five process model similarity measures we described and the data attribute measure. For the hierarchical clustering, we change the linkage between: single, complete, upgma, and wpgma;[5] and iterate the number of clusters for the following values: $\{2, 5, 10, 15, 20, 30, 40, 50, 100, 200\}$. Considering all combinations of an LPM set, distance extraction method, and clustering algorithm parameter, we have 1440 experiments in total. Due to space limitations, we only show the results of some of the experiments or a summary, but we upload all obtained data for the experiments to *Zenodo*[6] and the analysis notebooks to *Gitlab*[7] for interested readers.

5.3 LPM samples comparison

The focus of the paper is to showcase that a sample of representative LPMs are better suited to capture the underlying process than a sample of top-scoring LPMs. Since the capturing of the process' essence is an abstract utility, we quantify it using two measures that serve as proxies:

Table 2 Local process model sets used in the evaluation

Event Log	LPM set	Number of models
BPI Challenge 2012	$\text{LPM}_{\text{BPIC2012}}$	547
BPI Challenge 2012 - resource 10939	$\text{LPM}_{\text{BPIC2012-res10939}}$	562
BPI Challenge 2017 - Offers Log	$\text{LPM}_{\text{BPIC2017}}$	578
BPI Challenge 2020 - Prepaid Travel Costs	$\text{LPM}_{\text{BPIC2020}}$	292
Sepsis	$\text{LPM}_{\text{Sepsis}}$	505
Hospital Billing	LPM_{HB}	626

[4] https://github.com/promworkbench/LocalProcessModelDiscoveryByCombiningPlaces/releases/tag/jiis24-grouping-lpms

[5] For more information on the upgma and wpgma linkage, see https://haifengl.github.io/clustering.html.

[6] https://zenodo.org/records/17013367

[7] https://git.rwth-aachen.de/evaluations/jiis2024-groupinglpms

1. *Coverage* - the fraction of events covered by the full set of LPMs that are also covered by the sample. Given an event log L, an LPM set LPM, and a sample $\mathrm{LPM}_s \subseteq \mathrm{LPM}$, we compute coverage as $cov = \frac{|\{e \in s | s \in \lambda(\mathrm{lpm}, L) \wedge \mathrm{lpm} \in \mathrm{LPM}_s\}|}{|\{e \in s | s \in \lambda(\mathrm{lpm}, L) \wedge \mathrm{lpm} \in \mathrm{LPM}\}|}$.
2. *Diversity* - the mean pairwise distance between the LPMs in the sample to quantify model repetition. Given an LPM sample LPM and a distance function dist, we compute diversity as $div = \frac{\sum_{\mathrm{lpm}_1, \mathrm{lpm}_2 \in \mathrm{LPM}, \mathrm{lpm}_1 \neq \mathrm{lpm}_2} \mathrm{dist}(\mathrm{lpm}_1, \mathrm{lpm}_2)}{|\mathrm{LPM}| \cdot (|\mathrm{LPM}| - 1)}$.

Hence, this part of the evaluation compares the coverage and diversity of the top-scoring sample of LPMs to the computed representative samples based on distance and score, for each clustering.

Setup First, for each clustering result, we derive three samples: (1) sample containing the highest scoring LPMs in each of the resulting groups according to the $\mathrm{rank}_L^{\mathrm{freq}}$ ranking (*score representative sample* (RS)), (2) sample containing all the medoid LPMs of the resulting groups, or the highest scoring LPMs in each group according to the $\mathrm{rank}^{\mathrm{dist}}$ ranking (*distance representative sample* (RD)), and (3) sample containing the top *k* highest scoring LPMs overall according to the $\mathrm{rank}_L^{\mathrm{freq}}$, where *k* corresponds to the number of groups in the clustering (*top sample* (T)). Then, for each sample, we compute *coverage* and *diversity* and compare them pairwise.

Choosing any of the samples is deciding between the trade-offs associated with them. The T sample is expected to have high coverage, as it contains the most frequent LPMs, but as illustrated in the example in Section 1, some of its coverage may be duplicated. Moreover, since the sample is selected without considering similarity between the LPMs, it is expected to have low diversity. The RS sample is also expected to achieve high coverage, as it includes the most frequent LPMs in each group. Since the groups are formed based on similarity, the RS sample is also expected to have higher diversity than the T sample and mitigate duplicate coverage. The RD sample, including the medoid of each group, is expected to have high diversity. However, since the medoids are not selected based on frequency, the coverage of the RD sample is sacrificed twice: first, by grouping similar LPMs together, and second, by not selecting the most frequent LPM in each group. We expect that, in terms of coverage, the RS and T samples will always be better than RD, and will alternate between each other depending on an event log, the clustering parameters, and similarity measures. In terms of diversity, we expect the order to be RD, RS, and last T.

Results To compare the samples, we compute the difference between the computed coverage and diversity values for each pair of samples per obtained clustering. Considering that the bigger the sample, the top-scoring sample would have an advantage, we restrict to $k \in \{2, 5, 10, 15, 20\}$. The results are shown for each event log separately on Fig. 4, where the x-axis represents the computed difference values and the y-axis the different pairwise sample comparisons for both measures (in total six). From the figure, we can make the following conclusions.

The coverage computed for the RD sample is consistently lower than the coverage computed for the T and RS samples. This is visible from the orange and green boxplots. The orange boxplot represents $cov_{RD} - cov_T$ and shows that most values are negative. Thus,

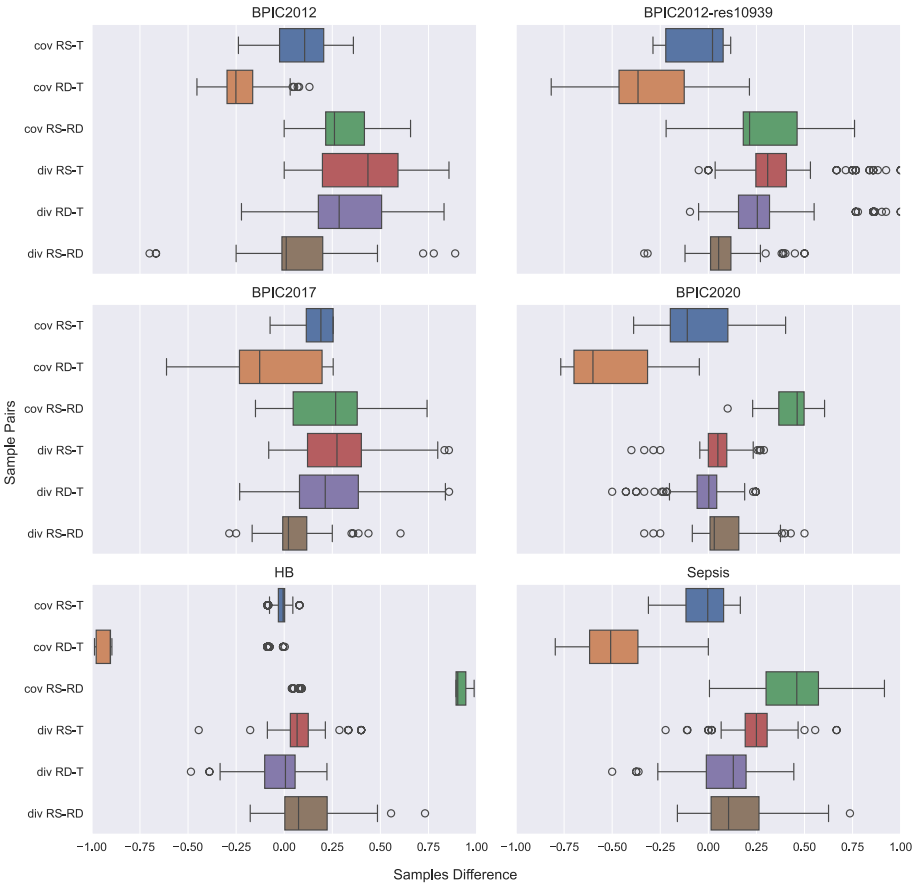


Fig. 4 The difference in coverage and diversity between the samples. The notation *cov S1-S2* represents values obtained using $cov_{S1} - cov_{S2}$, where *cov* stands for coverage, *div* for diversity, and *S*1, *S*2 are placeholders for T (top sample), RS (score representative sample), and RD (distance representative sample)

the coverage computed for the RD sample is in most cases lower than the coverage computed for the T sample. In contrast, the green boxplot represents $cov_{RS} - cov_{RD}$ and indicates that the majority of the observations fall into the positive range. Thus, the coverage computed for the RS sample is in most cases higher than the coverage computed for the RD sample. The coverage difference between T and RS samples, represented as a blue boxplot, varies between event logs. For the *BPIC* event logs, in most cases the RS sample has higher coverage, while for *HB*, *Sepsis*, and *BPIC2020* event logs it is almost half-half split between negative and positive values. All results surrounding coverage support our initial expectations.

The diversity is higher for almost all event logs for the RS and RD samples compared to the T sample (red and purple boxplots, respectively). Exceptions are the *HB* and *BPIC2020* event logs for the RD sample, where the values are equally distributed below and above 0, which is quite surprising. What is also surprising is that the RS samples score better on diversity than the RD samples (as indicated by the brown boxplots). A potential explana-

tion would be that the representatives in the RD sample try to balance the distance to all LPMs in the group, even those that also closely relate to LPMs from other groups. Hence, the medoids in each group sometimes also overlap with the medoids in the complete set, resulting in medoids that are close in distance. Our assumption about the RD sample having the highest diversity does not hold, making the RD sample inferior to the RS sample in both diversity and coverage.

5.4 Correlating cohesiveness to representativeness

In the previous section, we found that RS samples outperform RD samples. Considering that the clustering results can influence the samples, we investigate whether these results and improvements compared to T samples correlate with cluster cohesiveness. We assess the cohesiveness of the clustering results by computing the silhouette score (Rousseeuw, 1987). Most of the computed silhouette score values lie between -0.5 and 0.5 indicating overlapping clusters for most distance measures, clustering parameters, and LPM sets. The majority of the values above 0.5 occur for the TL and DV measures, while the extremely negative values occur for NM and single linkage. We could not clearly identify any other patterns.

Setup We consider three subsets of the data for which we analyze correlation: (1) all 1440 clustering, (2) clustering where number of clusters is not larger than 20, and (3) clustering where number of clusters is not larger than 20 and $div_{RS} - div_{RD} > 0$. For each subset, we compute the Pearson correlation coefficient between the silhouette score values and the difference values in coverage and diversity between the three samples we computed in Section 5.3.

Results None of the 18 obtained values show an indication of a strong correlation between cohesiveness and representativeness. However, for $cov_{RS} - cov_T$ and $cov_{RD} - cov_T$, and subsets (2) and (3) introduced above, the values are around 0.3, hinting a weak correlation. A significant jump in correlation is noticeable for the $div_{RS} - div_T$ and $div_{RD} - div_T$ values, from around 0 when all clustering are considered to almost 0.3 for clustering with not more than 20 clusters. This jump and the improvement from 0.2 to 0.3 for the coverage, shows that our decision to restrict to clustering with not more than 20 clusters in the previous evaluation was reasonable. All results obtained for clustering where $div_{RS} - div_{RD} > 0$ were very close to the results obtained for the subset containing clustering with no more than 20 clusters.

5.5 Execution time analysis

Here, we investigate the time complexity of grouping the LPMs and also relate it to the time needed for discovery.

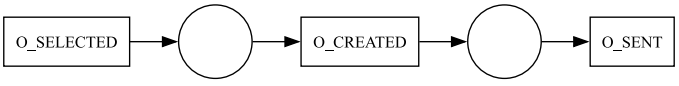
The execution time of the grouping framework depends mainly on two factors: (1) the distance computation between LPMs, and (2) the clustering. For the distance computation, 20 of the 36 computations are below 10 seconds, most of the rest are up to 20 seconds, one is 30 seconds, and one is 100 seconds. All above 10 seconds are measures requiring generating the language of the models, TM and EF, and the GED measure. The highest computation is for the DV measure and the *BPIC2017* event log. The clustering times are

all below 3 seconds. When comparing to the times needed for discovering the LPMs, in the majority of experiments (around 1100), the grouping including the distance computation between LPMs, takes less than half the time the discovery needs. However, for a smaller subset (around 150), the grouping can take up to three times the discovery time. It is worth noting that the reported times may vary from execution to execution. Therefore, to obtain more stable results, we need to let the experiments run a certain number of times and then produce aggregated values.

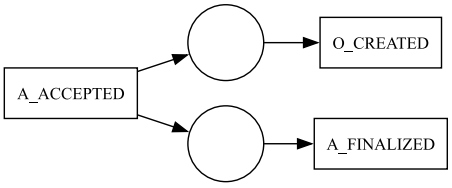
5.6 Case study: BPIC2012-res10939

To complement the quantitative analysis above, we finish with examining three representative models for the *BPIC2012-res10939* event log. The advantage of such a small case study is that it demonstrates the benefits of grouping from the user's perspective.

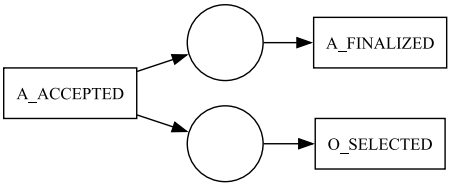
In Fig. 1, we showed that the top-scoring models for both (Tax et al., 2016b) and (Peeva et al., 2022) focus on different behavioral variants of *O_SELECTED*, *O_CREATED*, and *O_SENT*. Such repetition appears for lower-ranked models as well. In Fig. 5, we show the three top-scoring representative LPMs after grouping the original set of LPMs. To obtain the LPMs, we use the results for the EF measure with wpgma linkage and 20 clusters. It is clear that the collective language of these three models is larger than of those described by the three top-scoring original models discovered by both Peeva et al. and Tax et al. (see Fig. 1). Additionally, the distance between the three representative models is 0.92 while between the three top-scoring models is 0.18. The distance between all representatives is also 0.92 while between the 20 top-scoring models is 0.58. Moreover, the three top-scoring models cover 372 out of the 2763 events in the *BPIC2012-res10939*, while the three representative models cover 578 events. Even more interesting is that the 20 top-scoring models together cover also 578 events or the same as the top three representatives, while all representatives together cover 658 events. This combination is one example where grouping and selecting representatives improves both dimensions.



(a) LPM 1 - original rank 1



(b) LPM 2 - original rank 5



(c) LPM 3 - original rank 7

Fig. 5 The three top-scoring representative LPMs for *BPIC2012-res10939*

6 Discussion and future work

6.1 Limitations

The LPM grouping framework we presented, in theory, can be as flexible as possible regarding the representation of the allowed models, the way distance is measured, and clustering options. However, by offering a specific implementation of it, we made design decisions that restrict its potential and introduced different biases.

LPM representation In our framework, we restrict to LPMs represented as accepting Petri nets. However, LPMs can be depicted using any process modeling language as long as they represent patterns found in the data. Therefore, using a more different representation will also require adapting the distance measures. However, this is a limitation of the implementation, not of the framework.

Distance measures We offer two ways to measure the distance between LPMs: (1) structure and/or behavior and (2) the context in which they appear. For the process model similarity measures, we considered existing surveys and chose similarity measures representing the different categories in which the measures are divided. As part of future work, we can extend the framework with additional measures.

For groping LPMs based on the context in which they appear, we encode the context by using the event attribute values of events covered by an LPM. Specifically, we summarize the values for each attribute using several descriptive indicators depending on the attribute type. By limiting to a specific set of indicators, we introduce bias in how we represent the context and when two LPMs would be considered similar based on it. In future work, we can extend the offered descriptive indicators. However, the data attribute measure might be unusable for large-scale event logs with many attributes because of performance problems. Encoding categorical attributes using frequency vectors can result in sparse vectors, which in turn affects the complexity and robustness of the clustering. As an extension, the framework can support various heuristics for choosing valuable attributes and grouping infrequent categorical values. Furthermore, some activities can have more attributes than others, creating an imbalance in the context representation of LPMs that include these activities. More specifically, the context and, thereby, the created groups will more heavily depend on the context of one activity (the one with more attributes) compared to the rest of the activities in the LPM. Possible solutions would be to exclude non-overlapping attributes or use distance measures better suited for such situations.

Clustering We use hierarchical clustering as it does not require preselecting the number of clusters, is not dependent on many parameters, is more flexible regarding distance measures compared to other algorithms, and, depending on the linkage, it can find both spherical and non-spherical clusters. At the same time, hierarchical clustering is computationally heavy and might not be best suited for larger sets of LPMs. As part of future work, we can extend the framework with additional clustering algorithms and parameters.

6.2 Evaluation results

The evaluation we performed, while extensive, it is not comprehensive. LPM discovery algorithms discover hundreds or thousands of models for one event log. Hence, we can assume that the variation of possible patterns explodes further when we consider processes in different contexts, different discovery algorithms, and parameters. Hence, the experiments we performed can be extended as part of future work.

From our experiments, there are several conclusions we can make: (1) the clusters we get for the different event logs, distance measures, and parameters are mainly not cohesive, (2) despite obtaining non-cohesive clusters, the score representative sample computed from the grouping mostly scores better on the diversity measure and the coverage measure compared to when for each group the top-scoring LPM is chosen, (3) cohesiveness does not correlate with representativeness, and (4) a few distance computations are execution heavy. Considering this, there are several considerations for the future. First, we can perform a root cause analysis on the results to uncover the source of the poor cluster compactness. Second, we should extend the framework with additional measures than the diversity and coverage. With the current results, there was no indication that more cohesive clusters produce more representative samples. However, there is no sufficient data with high-quality clustering results to be confident in the obtained results. Therefore, can be explored further in future work.

7 Conclusion

In this paper, we described the challenges of model explosion and repetition in LPM discovery, and motivated and proposed a framework for grouping similar LPMs. We have two strategies to measure distance between two LPMs. On the one hand, we use established process model similarity measures from the literature. On the other hand, we consider the events an LPM covers, and use the data attributes available for those events to build an occurrence context of the LPM. Then, we compute the distance between two LPMs as the distance of the respective occurrence contexts. The proposed framework accepts an event log and a set of LPMs as input, and consists of three-steps. In the first one, we use the aforementioned distance notions to compute pairwise distances between the LPMs in the set. Then, the computed distances are used to cluster the set of LPMs, and for each cluster, one LPM is chosen as a cluster representative. In the evaluation, we showed how grouping similar LPMs together improves process understandability on a real-life case study. Finally, we showcased LPM diversity improvement on six real event logs for the original set of LPMs and the computed set of representative LPMs. We finished by discussing limitations of the implementation of the framework and the performed evaluation.

Acknowledgements We thank the Alexander von Humboldt (AvH) Stiftung for supporting our research. The authors gratefully acknowledge the financial support by the Federal Ministry of Education and Research (BMBF) for the joint project AIStudyBuddy (grant no. 16DHBKI016).

Author Contributions V.P. wrote the manuscript text and W.M.P v. d. A. reviewed the manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL. Alexander von Humboldt-Stiftung and Bundesministerium für Bildung und Forschung, 16DHBKI016.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Ethics Approval Not Applicable.

Competing Interests The authors declare no competing interests.

References

Acheli, M., Grigori, D., & Weidlich, M. (2019). Efficient discovery of compact maximal behavioral patterns from event logs. In *CAiSE 2019* (pp. 579–594).

Acheli, M., Grigori, D., & Weidlich, M. (2022). Discovering and analyzing contextual behavioral patterns from event logs. *IEEE Transactions on Knowledge and Data Engineering, 34*(12), 5708–5721.

Agrawal, R., & Srikant, R. (1995). Mining sequential patterns. In *ICDE 1995* (pp. 3–14).

Ahn, H., & Chang, T.-W. (2019). A similarity-based hierarchical clustering method for manufacturing process models. *Sustainability*, *11*(9).

Aiolli, F., Burattin, A., & Sperduti, A. (2011). A business process metric based on the alpha algorithm relations. In *BPM 2011 international workshops*, Revised Selected Papers, Part I (pp. 141–146).

Awad, A., Polyvyanyy, A., & Weske, M. (2018). Semantic querying of business process models. In *ECOC 2008* (pp. 85–94).

Becker, M., & Laue, R. (2012). A comparative survey of business process similarity measures. *Computers in Industry, 63*(2), 148–167.

Bergenthum, R., Desel, J., Lorenz, R., & Mauser, S. (2007). Process mining based on regions of languages. In *BPM 2007* (pp. 375–383).

Bergmann, R., Müller, G., & Wittkowsky, D. (2013). Workflow clustering using semantic similarity measures. In *KI 2013* (pp. 13–24).

Brunings, M., Fahland, D., & Verbeek, E. (2022). Discover context-rich local process models (extended abstract). In *ICPM-D 2022* (pp. 100–103).

Brzychczy, E., Aleknonyte-Resch, M., Janssen, D., & Koschmider, A. (2025). Process mining on sensor data: A review of related works. *Knowledge and Information Systems, 67*(6), 4915–4948.

Carmona, J., Cortadella, J., & Kishinevsky, M. (2008). A region-based algorithm for discovering Petri Nets from event logs. In *BPM 2008* (pp. 358–373).

Dalmas, B., Tax, N., & Norre, S. (2018). Heuristic mining approaches for high-utility local process models. *Transactions on Petri Nets and Other Models of Concurrency, 13*, 27–51.

Deeva, G., & Weerdt, J. D. (2018). Understanding automated feedback in learning processes by mining local patterns. In *BPM 2018 international workshops* (pp. 56–68).

Delcoucq, L., Lecron, F., Fortemps, P., & van der Aalst, W. M. P. (2020). Resource-centric process mining: Clustering using local process models. In *SAC 2020* (pp. 45–52).

Dijkman, R.M., Dumas, M., & García-Bañuelos, L. (2009). Graph matching algorithms for business process model similarity search. In *BPM 2009* (vol. 5701, pp. 48–63).

Dijkman, R.M., van Dongen, B.F., Dumas, M., García-Bañuelos, L., Kunze, M., Leopold, H., Mendling, J., Uba, R., Weidlich, M., Weske, M., & Yan, Z. (2013). A short survey on process model similarity. In *Seminal contributions to information systems engineering, 25 Years of CAiSE* (pp. 421–427).

Dijkman, R. M., Dumas, M., Reina Käärik, B. F., & Mendling, J. (2011). Similarity of business process models: Metrics and evaluation. *Information System, 36*(2), 498–516.

Dumas, M., García-Bañuelos, L., & Dijkman, R. M. (2009). Similarity search of business process models. *IEEE Data Engineering Bulletin, 32*(3), 23–28.
Ekanayake, C. C., Dumas, M., García-Bañuelos, L., Rosa, M. L., & Hofstede, A. H. M. (2012). Approximate clone detection in repositories of business process models. In *BPM 2012* (pp. 302–318).
Guzzo, A., Rullo, A., & Vocaturo, E. (2022). Process mining applications in the healthcare domain: A comprehensive review. *WIREs Data Mining Knowledge Discovery,12*(2).
Heinze, T.S., Amme, W., & Schäfer, A. (2021). Detecting semantic business process model clones. In *ZEUS 2021* (vol. 2839, pp. 25–28).
Jin, T., Wang, J., Rosa, M. L., Hofstede, A. H. M., & Wen, L. (2013). Efficient querying of large process model repositories. *Computers in Industry, 64*(1), 41–49.
Jung, J., & Bae, J. (2006). Workflow clustering method based on process similarity. In *ICCSA 2006, Proceedings, Part II* (pp. 379–389).
Kirchner, K., & Markovic, P. (2018). Unveiling hidden patterns in flexible medical treatment processes - A process mining case study. In *ICDSST 2018* (pp. 169–180).
Kuhn, H. W. (2010). The Hungarian method for the assignment problem. In *50 years of integer programming 1958-2008 - From the early years to the state-of-the-art* (pp. 29–47).
Leemans, M., & van der Aalst, W. M. P. (2014). Discovery of frequent episodes in event logs. In *SIMPDA 2014*, Revised Selected Papers (pp. 1–31).
Leemans, S. J. J., Fahland, D., & van der Aalst, W. M. P. (2013). Discovering block-structured process models from event logs: A constructive approach. In *PETRI NETS 2013* (pp. 311–329).
Leemans, S. J. J., Tax, N., & Hofstede, A. H. M. (2018). Indulpet miner: Combining discovery algorithms. In *OTM 2018* (pp. 97–115).
Li, H. (2014). Smile. https://haifengl.github.io.
Mannel, L. L., & van der Aalst, W. M. P. (2019). Finding complex process-structures by exploiting the token-game. In *PETRI NETS 2019* (pp. 258–278).
Mannhardt, F., & Tax, N. (2017). Unsupervised event abstraction using pattern abstraction and local process models. In *RADAR+EMISA 2017* (pp. 55–63).
Mannhardt, F., Leoni, M., Reijers, H. A., van der Aalst, W. M. P., & Toussaint, P. J. (2018). Guided process discovery - A pattern-based approach. *Information Systems, 76*, 1–18.
Mannila, H., Toivonen, H., & Verkamo, A. I. (1997). Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery, 1*(3), 259–289.
Munoz-Gama, J. J. M., et al. (2022). Process mining for healthcare: Characteristics and challenges. *Journal of Biomedical Informatics, 127*, 103994.
Ordóñez, A., Ordoñez, H., Corrales, J. C., Lozada, C. A. C., Wives, L. K., & Thom, L. H. (2017). Grouping of business processes models based on an incremental clustering algorithm using fuzzy similarity and multimodal search. *Expert Systems with Applications, 67*, 163–177.
Peeva, V., & Aalst, W. M. P. (2023). Grouping local process models. In Smedt, J. D., & Soffer, P. (Eds.) *Process mining workshops - ICPM 2023*, Revised Selected Papers (pp. 419–430).
Peeva, V., Mannel, L. L., & van der Aalst, W. M. P. (2022). From place nets to local process models. In *PETRI NETS 2022* (pp. 346–368).
Pijnenborg, P., Verhoeven, R., Firat, M., Laarhoven, H.V., & Genga, L. (2021). Towards evidence-based analysis of palliative treatments for stomach and esophageal cancer patients: A process mining approach. In *ICPM 2021* (pp. 136–143).
Qiao, M., Akkiraju, R., & Rembert, A. J.: Towards efficient business process clustering and retrieval: Combining language modeling and structure matching. In *BPM 2011* (pp. 199–214).
Rosa, M. L., Dumas, M., Uba, R., & Dijkman, R. M. (2013). Business process model merging: An approach to business process consolidation. *ACM Transactions on Software Engineering and Methodology, 22*(2), 11–11142.
Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics, 20*, 53–65.
Sarno, R., Ginardi, R. V. H., Pamungkas, E. W., & Sunaryono, D. (2013). Clustering of ERP business process fragments. In *IC3INA 2013* (pp. 319–324).
Schoknecht, A., Thaler, T., Fettke, P., Oberweis, A., & Laue, R. (2017). Similarity of business process models - A state-of-the-art analysis. *ACM Computing Surveys, 50*(4), 52–15233.
Srikant, R., & Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. In *EDBT 1996* (vol. 1057, pp. 3–17).
Sun, S., Kumar, A., & Yen, J. (2006). Merging workflows: A new perspective on connecting business processes. *Decision Support Systems, 42*(2), 844–858.
Tax, N., Sidorova, N., van der Aalst, W. M. ., & Haakma, R. (2016a). Heuristic approaches for generating local process models through log projections. In *IEEE SSCI 2016* (pp. 1–8).

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

Tax, N., Dalmas, B., Sidorova, N., van der Aalst, W. M. P., & Norre, S. (2018). Interest-driven discovery of local process models. *Information Systems, 77*, 105–117.

Thaler, T., Schoknecht, A., Fettke, P., Oberweis, A., & Laue, R. (2016). A comparative analysis of business process model similarity measures. In *BPM 2016 international workshops* (pp. 310–322).

van der Aalst, W. M. P. (2020). On the pareto principle in process mining, task mining, and robotic process automation. In *DATA 2020* (pp. 5–12).

van der Aalst, W.M.P. (2016). *Process mining - data science in action*, Second Edition.

van Zelst, S.J., & Cao, Y. (2020). A generic framework for attribute-driven hierarchical trace clustering. In *BPM 2020, Revised Selected Papers* (pp. 308–320).

van Zelst, S. J., van Dongen, B. F., van der Aalst, W. M. P., & Verbeek, H. M. W. (2018). Discovering workflow nets using integer linear programming. *Computing, 100*(5), 529–556.

Wen, L., van der Aalst, W. M. P., Wang, J., & Sun, J. (2007). Mining process models with non-free-choice constructs. *Data Mining and Knowledge Discovery, 15*(2), 145–180.

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

Yan, Z., Dijkman, R. M., & Grefen, P. W. P. J. (2012). Business process model repositories - Framework and survey. *Information and Software Technology, 54*(4), 380–395.