An Experimental Evaluation of Process Concept Drift Detection

Jan Niklas Adams
RWTH Aachen University, Germany
niklas.adams@pads.rwth-aachen.de

Cameron Pitsch
RWTH Aachen University, Germany
cameron.pitsch@rwth-aachen.de

Tobias Brockhoff
RWTH Aachen University, Germany
brockhoff@pads.rwth-aachen.de

Wil M.P. van der Aalst
RWTH Aachen University, Germany
wvdaalst@pads.rwth-aachen.de

ABSTRACT

Process mining provides techniques to learn models from event data. These models can be descriptive (e.g., Petri nets) or predictive (e.g., neural networks). The learned models offer operational support to process owners by conformance checking, process enhancement, or predictive monitoring. However, processes are frequently subject to significant changes, making the learned models outdated and less valuable over time. To tackle this problem, *Process Concept Drift* (PCD) detection techniques are employed. By identifying when the process changes occur, one can replace learned models by relearning, updating, or discounting pre-drift knowledge. Various techniques to detect PCDs have been proposed. However, each technique's evaluation focuses on different evaluation goals out of accuracy, latency, versatility, scalability, parameter sensitivity, and robustness. Furthermore, the employed evaluation techniques and data sets differ. Since many techniques are not evaluated against more than one other technique, this lack of comparability raises one question: *How do PCD detection techniques compare against each other?* With this paper, we propose, implement, and apply a unified evaluation framework for PCD detection. We do this by collecting evaluation goals and evaluation techniques together with data sets. We derive a representative sample of techniques from a taxonomy for PCD detection. The implemented techniques and proposed evaluation framework are provided in a publicly available repository. We present the results of our experimental evaluation and observe that none of the implemented techniques works well across all evaluation goals. However, the results indicate future improvement points of algorithms and guide practitioners.

PVLDB Reference Format:
Jan Niklas Adams, Cameron Pitsch, Tobias Brockhoff, and Wil M.P. van der Aalst. An Experimental Evaluation of Process Concept Drift Detection. PVLDB, 16(8): 1856 - 1869, 2023.
doi:10.14778/3594512.3594517

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/cpitsch/cdrift-evaluation/.

Proceedings of the VLDB Endowment, Vol. 16, No. 8 ISSN 2150-8097.
doi:10.14778/3594512.3594517

1 INTRODUCTION

Processes are – either implicitly or explicitly – omnipresent in today's world. The use of administrative business processes [19], e.g., order-to-cash, or production processes [31, 45] is widespread in companies. The execution of such processes leaves data traces in the underlying databases of information systems supporting these processes [62]. These data traces are extracted in the form of an *event log* describing the process executions and their associated data. Starting from an event log, one learns different models, e.g., predictive models [11] or process models [28]. Subsequently, these models are used for the operational support of ongoing process executions. Using predictive models, one may uncover ongoing process executions leading to undesirable outcomes [26], long execution times [60], or problematic constraints [36]. Using a learned process model, one may check for conformance of ongoing executions [14] or calculate performance metrics [10] to provide actions for improved performance [51]. Such modern forms of operational support use algorithms and concepts from *process mining* [62].

However, processes seldom remain stable. In fact, processes are subject to significant changes caused by internal or external sources [2]. Such significant changes are called *Process Concept Drifts* (PCDs) [48]. Prominent examples are changes that were invoked by the past Covid epidemic or recent supply-chain shocks. However, not only events of global reach invoke concept drift but also local legislative changes, business realignments, or employee turnover. Such drifts are consequential for operational support: Learned and represented knowledge decays in value when not updated [21]. With a drift in the process, the reality also drifts further apart from the learned model, leading to inaccurate or even misleading operational support. Therefore, PCDs need to be detected such that a model can either be relearned, updated, or pre-drift knowledge can be discounted [16, 32].

Detecting process concept drifts is a challenging problem: Only the event data collected during the execution of the process are available. First, a representation has to be found that sufficiently encodes the structure of the process and will reflect significant changes while only using the underlying event data. Second, a concept drift detection algorithm has to be employed to this representation, classifying the correct point in time as concept drift. Several PCD techniques have been proposed in the context of process mining [48]. However, consensus about the exact problem definition and, therefore, targeted evaluation goals and employed evaluation metrics are missing. Papers use a subset of evaluation goals such as accuracy, latency, versatility, scalability, parameter sensitivity, and robustness. Additionally, an experimental evaluation of state-of-the-art PCD detection techniques is missing. The different proposed PCD detection techniques cannot be compared in a fair and systematic way. Therefore, guidance for two groups of people is absent: On the one hand, researchers can not accurately assess which state-of-the-art method is best suited for which purposes, making a selection of evaluation baselines and continued research difficult. On the other hand, practitioners can not accurately assess the best algorithm for their specific use case.

In this paper, we provide an evaluation framework and perform an experimental evaluation of process concept drift techniques. Our contributions are the following:

- We comprehensively collect evaluation goals and corresponding evaluation techniques used throughout the literature to compare PCD detection techniques.
- We provide a taxonomy of PCD detection techniques and select a representative subset of techniques for our framework.
- We provide a unified framework to compare PCD techniques in all relevant evaluation goals.
- We implemented the selected PCD detection and evaluation techniques in a unified Python framework to ensure comparability and present the results of this evaluation.

We introduce concept drift in process mining in Section 2 and define the problem of PCD detection in Section 3. Section 4 provides a taxonomy of PCD detection techniques and our representative sample of algorithms. We derive and define our evaluation framework in Section 5. We present and discuss our evaluation results in Section 6 and conclude this paper in Section 7.

2 BACKGROUND

Process mining is an emerging discipline connecting data science and process science. Starting from an event log extracted from an information system, process mining techniques learn implicit or explicit process models to provide descriptive [5], predictive [11, 43], or prescriptive [69] insights for the underlying process. These techniques are typically used for operational support, with regular deployment in industry. The general problem of concept drifts in data is known for a long time [49], therefore, the problems of dealing with concept drift affecting the accuracy of data-driven process insights were already formulated more than a decade ago [64].

Since concept drift is not a problem exclusive to processes, many algorithms have been proposed to detect drift in generic data sequences [32] influencing the development of PCD detection techniques. Algorithms are classified into three categories: Error rate-based [20], data distribution-based [25], and multiple hypothesis test drift detection [4]. In error rate-based drift detection, a model is learned on historical data and tested against new data points [7]. If the error is large enough, a drift is classified. The error depends on the window size of included data points, therefore, ADWIN [6] has been introduced to automate window size adaptation. In data distribution-based drift detection, data populations of two different windows are quantitatively compared [52, 54]. If the distributions are significantly different, a concept drift is detected. As the name indicates, multiple hypothesis testing drift detection techniques perform several detections to get more support for classifications, i.e., an ensemble classifier [18], or a hierarchical classifier [3].

The problem of detecting concept drifts in a process vs. detecting concept drifts in a data stream is difficult for one reason: there are no direct numerical representations of the actual (unknown) process that could be compared. Starting from the underlying event data, techniques, first, have to derive some features describing process behavior and capturing *first-order process dynamics*. Second, these features have to be summarized into one global process representation. Third, process representations have to be investigated for significant changes over time, i.e., *second-order process dynamics*. The problem of deriving and constructing intermediate features to derive numerical representation for nun-numerical phenomena is analogously encountered in other areas, such as video concept drift [58]. We use the three steps of process feature extraction, process representation, and drift assessment to horizontally categorize PCD detection techniques in Section 4 and derive a representative sample of techniques to be used in our experimental evaluation.

Most of the introduced techniques focus on detecting concept drifts in the structure of the process, i.e., the control flow. Several papers address problems beyond control-flow concept drift detection. Adams et al. [2] introduce a framework extracting a time series representation of other process dimensions (performance, data, or resources) to detect and correlate concept drifts. Ostovar et al. [41] introduce a technique to not only detect a concept drift but also characterize the underlying change. Concept drift characterization was also tackled by other researchers [50, 70].

Furthermore, different concept drifts types are considered in the literature. The most prominent type is *sudden drift*, a drift that occurs instantaneously. Several papers consider drift types such as *gradual drifts, recurring drifts, incremental drifts,* or *multi-order drifts* [35]. We limit our evaluation to sudden drifts for the following reasons: Sudden drifts are the most studied drifts and the detection of almost all other drift types can be reduced to the detection of sudden drifts. Gradual and incremental drifts are detected by leaving a gap in a time series and testing for sudden drifts [35]. Recurring drifts can be detected by individually detecting and combining them later. Furthermore, multi-order drifts are studied in a very limited way and, therefore, excluded in our framework.

A unified framework to evaluate PCD techniques is so far missing. The comparative work on PCD detection techniques is, generally, sparse. Sato et al. [48] provide an extensive literature review comparing and classifying different PCD detection techniques. However, they do *not* quantitatively compare techniques. Ceravolo et al. [15] present an evaluation specifically tailored to online PCD detection techniques across multiple evaluation goals, namely memory consumption, number of algorithm executions, latency, and detected drifts. However, the focus on online settings neglects important algorithm quality dimensions such as the versatility of the algorithm, i.e., which change patterns can be detected, the sensitivity to parameters and noise, and the scalability of the algorithms. Omori et al. [39] compare two PCD detection techniques across multiple dimensions and evaluate their applicability.

When looking at the evaluations of newly-introduced PCD detection techniques, even a consensus on the evaluation goals and metrics is missing: Each technique is evaluated according to an ad hoc selection of evaluation goals. However, an evaluation across all relevant evaluation goals is necessary to objectively compare PCD

Table 1: Evaluation goals covered by different newly introduced PCD detection techniques and evaluations.

Related work	EG_1	EG_2	EG_3	EG_4	EG_5	EG_6
[34, 35, 42]	✓	✓	✓	✓	✓	✓
[72]	✓	✓	✓	✓	✓	
[30]	✓	✓	✓	✓		
[12, 50]	✓	✓	✓			
[8, 33, 56, 57]	✓					
[22]	✓	✓		✓		✓
[9, 23, 61]	✓					✓
[39]	✓				✓	✓
[1]	✓		✓		✓	
[59]	✓		✓			
[70]	✓		✓			✓
[15, 71]	✓		✓	✓		✓
[38]	✓			✓		✓
[24]	✓			✓	✓	✓
[29]	✓		✓	✓	✓	✓
[13]			✓	✓		
[46]		✓		✓		✓
$\sum$	24	9	14	13	8	15

detection techniques. An evaluation goal is relevant if it was considered in a significant number of evaluations. We analyzed each evaluation of a PCD detection technique for the targeted evaluation goals. In total, we found six relevant evaluation goals:

Quality of Results:

- EG_1 **Accuracy**: Correspondence between detected and actual change points
- EG_2 **Latency**: Closeness of detected to actual change points
- EG_3 **Versatility**: Coverage of different change patterns

Quality of Algorithm:

- EG_4 **Scalability**: Computation time
- EG_5 **Parameter Sensitivity**: Impact of algorithmic settings
- EG_6 **Robustness**: Impact of data quality issues on the results

Table 1 depicts the evaluation goals targeted by each evaluation of a PCD detection technique. Since all evaluation goals show sufficient support, we include all evaluation goals in our framework.

Different evaluation techniques are available for each evaluation goal. We introduce these techniques along with our selection in Section 5. However, we first provide a taxonomy of PCD detection techniques to choose a representative set of techniques used for the initial population of our framework in Section 4. Our taxonomy, publicly available implementations, and selection are depicted in Table 2. Other techniques, event logs, or evaluation goals can easily be added to the publicly available implementation.

3 PRELIMINARIES

An event log contains the event data collected throughout the execution of a process. Each event is associated with an activity describing the executed action and a timestamp describing the time of execution. Different events belonging to the same process execution are grouped together through a common case.

Definition 1 (Event Logs). *We denote the universe of event identifiers with $\mathcal{E}$, the universe of case identifiers with $\mathcal{I}$, and the universe of activities with $\mathcal{A}$. Each event is associated with an activity $\pi_{act} : \mathcal{E} \to \mathcal{A}$ and timestamp expressed as natural number $\pi_{time} : \mathcal{E} \to \mathbb{N}$. Furthermore, each event is associated to a case $\pi_{case} : \mathcal{E} \to C$. An event log is a subset $E \subseteq \mathcal{E}$ of events grouped as event sequences per case $\pi_{cases}(E) = \{\langle e_1, \dots, e_n \rangle \in E^* \mid \exists_{id \in \mathcal{I}} \{e_1, \dots, e_n\} = \{e \in E \mid \pi_{case}(e) = id\} \wedge \pi_{time}(e_1) < \cdots < \pi_{time}(e_n)\}$.*

Each event log contains a ground truth of concept drifts in the form of a subset of timestamps. This includes the empty set, i.e., no concept drift being present.

Definition 2 (Process Concept Drifts). *Let $E \subseteq \mathcal{E}$ be an event log. The ground truth of concept drifts contained within E is denoted with $c(E) \subseteq \{\pi_{time}(e) \mid e \in E\}$.*

An example would be a delivery process. In a pre-drift period, the process exhibits sequential behavior for sending an order, i.e., first, collecting the items and, subsequently, preparing the packages. At some point in time, the process is subject to a concept drift: preparing the packages and collecting the items are now concurrent. This drift has consequences for operational support, e.g., remaining time prediction might be significantly lower when factoring in concurrency instead of sequentiality. In reality, we do not have explicit process models to easily spot changes, we only have the event log that records the sequence of activities for each customer.

A process concept drift detection technique maps an event log to a set of hypothesized concept drift change points.

Definition 3 (Process Concept Drift Detection). *Let $E \subseteq \mathcal{E}$ be an event log. Let $\mathcal{P}$ be a parameter space and $p \in \mathcal{P}$ be a parameter setting of this space. A concept drift detection technique $d_p(E) \subseteq \{\pi_{time}(e) \mid e \in E\}$ proposes change points of concept drifts that should correspond to the ground-truth concept drifts $c(E)$.*

4 PROCESS CONCEPT DRIFT DETECTION

We give an overview of existing PCD approaches from which we select a subset for our evaluation. We structure the existing works using a taxonomy for *process* concept drift detection approaches. In contrast to existing taxonomies on *General Concept Drift Detection* (GCDD) such as [21], we particularly focus on the process aspect with event data being the common input. Our taxonomy is based on three main challenges every PCD approach must address:

(1) How is the information from a single process execution represented — that is, which *process-centric features* are extracted?
(2) Given the features, how is the status quo—namely, the *process—represented*?
(3) Based on the representation, how are potential *drifts assessed* (i.e., measure and quantify)?

Process features capture observed, usually local, behavior in single process executions. Consequently, combining and aggregating features extracted from multiple process executions constitutes the *process representation.* This representation is used to assess if a given set of process executions describes a changed process.

Compared to the classification used by Sato et al. in their detailed overview work on PCD [48], we build on commonalities rather than maximal differing characteristics. For example, since many

Table 2: Classification of different PCD detection techniques according to our taxonomy. Techniques with publicly available implementation are italicized. Techniques implemented and included in our experimental evaluation are bolded.

			Process Change Assessment			
			Feature-centric		**Feature Space Description**	
			Feature Population	**MV Feature Vector Series**	**Process Model**	**Process Model-free**
Process Features	**Activity Pairs**	**local**	***J-Measure, Window Count, Bose 2011*** [8] ***J-Measure, Window Count, Bose 2014*** [9] Kumar 2015 [61] ***ADWIN J, ADWIN WC, Martjushev 2015*** [38]	Accorsi 2011 [1] Richter 2017 [46] ***RINV, Zheng 2017*** [72] ***LCDD, Lin 2020*** [29] *Yeshchenko 2021* [70, 71]		Lakshmanan 2011 [27]
		global	***J-Measure, Window Count, Bose 2011*** [8] ***J-Measure, Window Count, Bose 2014*** [9] ***ADWIN J, ADWIN WC, Martjushev 2015*** [38] *Ostovar 2016* [42]	Hassani 2019 [22]		*Hompes 2015* [23]
	Multiple Act.	**loc.**	**EMD, Brockhoff 2020** [12]		Liu 2018 [30] Stertz 2018 [56], 2019 [57] *Tavares 2019* [59]	Luengo 2011 [33] Carmona 2012 [13]
		glo.	***PRODRIFT, Maaradji 2015*** [34], ***2017*** [35] ***PGM, Seeliger 2017*** [50]	Hassani 2019 [22] Impedovo 2020 [24]		

approaches use statistical hypothesis tests, they constitute a top-level class in [48]. In contrast, we rather horizontally categorize approaches considering the common challenges. We do not aim to categorize the field of PCD as a whole but rather focus on the evaluation of existing approaches in a restricted setting—that is, offline, sudden drift detection. Therefore, our taxonomy focuses on key algorithmic concepts.

Process Features. A major focus of existing works usually lies on engineering robust but yet expressive process features. If certain behavior such as long-term dependencies is not captured by the features, certain changes can not be detected. However, complex and intrinsically high-dimensional features suffer from the curse of dimensionality and are less robust to noise.

On a high level, process features can be classified by their *scope* and the required *extraction context*. Each of these dimensions are shown in the rows of Table 2.

Regarding the *feature scope*, we further distinguish between features that cover pairs of activities or longer subtraces. As can be seen in the first two rows of Table 2, showing the classification of existing works, most works consider relations over activity pairs For example, such relations describe whether activities directly or eventually follow each other [8, 9, 29, 71, 72].

The *extraction context* describes whether a feature can be extracted from a single case (local) or whether multiple cases must be considered (global). While we can extract directly-follows relations per case, measures such as the J-measure [8], which quantifies the quality of a rule, require considering the entire event log. In the extreme, multi-activity case, the trace variant itself becomes a local feature [12], or a global symbolic representation is assigned to a group of traces [34, 35]. We decided to consider the extraction context as a separate aspect because it captures how much process information is already contained in the process feature.

An interesting variant of global multi-activity features are process models discovered or features extracted from sublogs. Adjacent models in a multivariate model series are then, for example, compared using graph edit distance [22]. Even though process models play an important role in the domain of process mining, relatively few works use features extracted from a process model but rather directly use the low-level features (e.g, directly-follows relations) that are also used by many discovery algorithms.

Process Representation and Drift Assessment. Complementary to the feature extraction, we distinguish two more classes in our taxonomy that focus on the remaining two aspects of *process concept drift detection*–namely, *process concept* and *drift detection.* To eventually detect concept drift, every method must have an internal representation of the current *concept* describing the status quo. Based on this representation, potential drifts can be measured. We explicitly distinguish between drift assessment and localization. Many existing works do not focus on end-to-end detection, eventually returning the precise onsets of drifts, but rather return a series of drift severity scores over time [8, 12, 27, 61]. Table 2 shows the aspects of *process representation* and *drift assessment* in all the main forms observed in the literature. Process representations can be feature-centric or based on feature space descriptions. In the former case, the process is directly characterized by aggregated features (e.g., a bag of features). In contrast, some approaches describe the feature space. For example, using traces as features, a discovered model abstracts from the input and describes the space of accepted traces [30, 56, 57]. This might increase robustness w.r.t. noisy features but can also be imprecise allowing for additional behavior.

The classification in Table 2 shows that current works most frequently represent processes by means of feature populations or series of feature vectors. While these representations are very similar, the difference is that feature populations are unordered describing longer process segments, often extracted from sliding windows. Process model-based as well as feature space descriptive approaches are less frequently used. The *drift assessment* is tightly coupled with the representation. Confronted with new event data (e.g., the next event, next case, or, when using sliding windows, the process contained in the next window), it quantifies potential changes. While for feature populations established techniques such as (non-parametric) hypothesis tests exist, model-based representations require solutions more specific for process mining. For example, conformance-checking approaches measure differences and, consequently, changes between the model and the event data [30, 56]. Approaches build on multivariate feature vector series either employ existing detection techniques such as PELT [71] or custom scoring functions [72]. The latter is also used for process model-free feature space descriptions [13].

Comparing our proposed taxonomy to systematic reviews on GCDD methods such as [32], both distinguish similar means of data representation and drift assessment. Works in both fields frequently represent data by feature populations, probability distributions, or models. Consequently, similar methods are used to measure drift—namely, hypothesis tests, difference measures for probability distributions, or error functions that quantify the quality of the currently employed model. However, while [32] specifically emphasizes providing statistical bounds, and data modeling is considered optional; works on PCD particularly focus on modeling the complex input data. Therefore, our taxonomy puts an emphasis on the challenging task of data—process—modeling. Despite this challenge, the broader field of GCDD offers interesting new approaches—for example, based on multiple hypothesis tests or ensemble models [18]—that have not yet been investigated for PCD.

Given our classification of the process concept drift detection approaches, we selected the approaches for our evaluation according to the following criteria: First, we aimed for good coverage of different process features, representations, and drift assessment methods. Second, the code should be available to verify the correctness of our reference implementation. Furthermore, we focused on works for offline drift detection. Therefore, the approaches using model-based process representations were discarded because they considered a streaming setup. Moreover, the works that employed a model-free space description either focused on trace clustering [23, 33], or the code was not available [13, 27]. Thus, they were not considered in Section 6. Besides, we particularly included methods that were used as benchmarks in other works. The final list of implemented approaches includes [8, 9, 12, 29, 34, 35, 50, 72].

WINDOW COUNT & J MEASURE: Bose et al. [8, 9] propose activity pair-based features called *Window Count* and *J-Measure*. For a pair (a, b) of activities and a trace, WINDOW COUNT describes how many sequences of given length start with a eventually followed by b. Similarly, the J-Measure, introduced by Smyth and Goodman [53], scores the quality of the rule: if a happens, then b will eventually follow. The authors use a fixed-size sliding window and extract the features either locally for each trace or globally from the entire window. The resulting populations are then compared by means of non-parametric two-sample hypothesis tests (Kolmogorov-Smirnov or Mann-Whitney U-Test). The resulting p-values are manually inspected for drifts.

ADWIN J & ADWIN WC: Martjushev et al. [38] identified the window size as one central weakness of WINDOW COUNT & J MEASURE. To automate the visual inspection of the results and speed up the algorithm they introduced an adaptive window size. They do so by recursively applying hypothesis tests on smaller windows around a low p-value, drilling down into the exact location of the drift. For timeframes with very high p-values, the adaptive window size grows, allowing them to skip large segments.

EMD: Brockhoff et al. [12] also employ a sliding window. In their work, each trace in a window directly corresponds to a feature, which we consider a *multi-activity, local* feature. The process described in a window is then represented as a distribution over trace variants. In contrast to most works that represent processes by means of feature populations, no hypothesis test is applied but differences between distributions are *measured* using the Earth Mover's Distance. In doing this, a sequence of distances is computed, and, finally, by manual inspection of the signal, the change points are identified as local maxima in the graph.

PGM: Seeliger et al. [50] extract *global process graph* features from process models discovered by the Heuristics Miner [68]. In particular, the frequencies of edges and nodes in the process graphs are considered. Applying an (adaptive) sliding window, a *feature population* is extracted for each window. Adjacent windows are then compared using a G-test. If the resulting p-values are below a given threshold, a change point is reported.

PRODRIFT: Maaradji et al. [34, 35] propose to convert traces into so-called *runs*, which are partial orders of activities. A run is constructed from a totally ordered trace by relaxing the ordering of activities that were found to be concurrent. In their work, they identify concurrency using α-relations [63]. To this end, multiple traces must be considered, and therefore, we classify runs as *global* feature over *multiple activities*. Applying a sliding window, they extract a population of runs for each window. Eventually, these populations are compared by means of Chi-Square tests.

RINV: Zheng et al. [72] detect drifts based on a boolean *relation matrix*. For each case and each *pair of activities*, an entry in the matrix is set if and only if the activities are directly following each other in the case. Likewise, there are additional entries for eventually-follows relationships. Zheng et al. call a relation stable if it is either one or zero for a long period of time (i.e., a minimum number of adjacent entries, i.e., cases that subsequently started). Conceptually, we classify this representation as a series of feature vectors over activity relations within a case. Since relation stability is defined based on adjacent entries the order of the vectors matters. To detect drifts, first, change point candidates, i.e., points with stability change, are extracted. The change points candidates are then clustered using DBSCAN. Finally, change points correspond to cluster centroids with a minimal distance from each other.

LCDD Lin et al. [29] detect drifts based on changes in directly-follows relations. To this end, they maintain two windows: a static so-called *complete window* that does not move and a sliding *detection window*. When determining the size of the *complete window*, the authors opt for local directly-follows completeness—that is, all directly-follows relations of the currently active process are present. They provide a statistical motivation for a baseline window size and propose an iterative, adaptive approach that grows the window if new relations are observed. If the two windows start to differ w.r.t. a directly-follows relation, a drift is reported, and the *complete window* is shifted. To increase robustness, they also propose that changes must persist for an extended period of time.

Implicit/Embedded PCD Detection. The presented techniques of our taxonomy explicitly detect concept drifts in processes. However, several other techniques perform an implicit or embedded concept drift detection as part of concept drift adaptation for predictive process monitoring. In these approaches, the change is mostly assessed through accuracy losses of employed machine learning models [17, 37, 47]. Maisenbacher and Weidlich [37] investigate different incremental learners, while Palm et al. [47] and Chamorro et al. [17] focus on retraining strategies, i.e., which data to include. Chamorro et al. furthermore, present a technique that, first, explicitly detects a concept drift using J-MEASURE [8] and, second, relearns a model with post-drift traces. All of the approaches use multiple, local activities as part of their feature encoding.

5 EVALUATION FRAMEWORK

In this section, we introduce our evaluation framework for PCD detection techniques. First, we discuss event logs and parameters employed in our framework. Second, we provide an algorithm to assign detected to ground-truth change points. Third, we investigate the related work's evaluation techniques for each evaluation goal and formally define our chosen evaluation metrics.

5.1 Evaluation Data

Newly-introduced PCD approaches are often evaluated on different event logs making approaches incomparable. We collect the event logs employed in the evaluations of different PCD detection techniques. As missing adoption of event logs in other evaluations might point to issues of missing generalizability, we filter out event logs that have only been employed in one paper's evaluation [1, 22, 24, 30, 39–41, 57, 72]. The remaining event logs are included in our framework if two requirements are met: First, the event log needs to be publicly available. Second, there must be a clearly defined ground truth of concept drifts. We depict an overview in Table 3. All available real-life event logs have no completely known ground truth. With ground-truth concept drifts only partly being known, using such event logs will introduce skewed evaluation results: Unknown but present ground-truth drifts might be detected by a PCD detection technique, however, this detection would be classified as false positive. In the extreme case, an evaluation would indicate a poor performance for a technique that can detect any concept drift and a perfect performance for a technique that is only able to detect the known ground-truth drifts. Therefore, we do not include any real-life event logs.

To show the suitability of our event log selection for the intended evaluation, we will introduce the necessary criteria of the event log selection for each evaluation goal and show that our selection fulfills these. To evaluate accuracy, latency, and parameter sensitivity, our event logs need to contain sufficiently many ground-truth drifts such that our evaluation results can be attested with statistical significance. Our set of logs contains 132 drifts, creating a foundation to differentiate between PCD detection techniques. For versatility, our event logs need to cover a balanced distribution over many change types. In Figure 2, we depict the 16 change patterns [67] included in our event log selection, along with the relative distribution. For scalability, we need a sufficiently large number of data points (events) in our selection. The event logs comprise 4508965 events. To assess robustness, the event log selection should contain different explicitly quantified noise levels and different types of noise. Our selection contains noise from adding unfitting events and removing fitting events, both with different levels. Therefore, the selection allows for the assessment of noise impact.

Table 3: Assessment of event logs employed in evaluations.

Log	Ground truth available?	Publicly Available?
Bose 2011 [8]	✓	✓
Ostovar 2016 [42]	✓	✓
Ceravolo 2020 [15]	✓	✓
Maaradji 2015 [34]	✓	
[44, 55, 65, 66]		✓

5.2 Parameter Choice

All considered algorithms require parameter choices. Therefore, a strategy to select parameter values in our evaluation is needed. Sampling all parameter values is not realistic as many parameters have an infinite range. We follow the predominant strategy for choosing parameter values: equidistant sampling in the parameter space [24, 29, 34, 35, 42, 72]. For each of the approaches, we use a parameter sample guided by the author's recommended parameter setting or by their parameter sets during evaluation. As not all approaches are evaluated in this way, we need a default number of parameter samples to include. On average, 7.35 parameter values are sampled across evaluations of our employed techniques. If not specified otherwise, we sample 7 values for a parameter using an equidistant sampling. If there is more than one parameter, we sample all parameter value combinations. Where necessary, we employ a peak-finding algorithm to automate visual inspection.

5.3 Change Point Assignment

To classify detected concept drifts as true positives, detected and ground-truth drifts need to be linked under consideration of two questions: First, which time lag is acceptable for a detected drift w.r.t. the ground-truth drift? Second, how to deal with multiple detected change points around the same ground-truth change point? We propose a bipartite matching of detected change points and actual change points given some allowed lag. For an acceptable lag window $lag \in \mathbb{N}$, a parameter setting $p \in \mathcal{P}$, and an event log $E \subseteq \mathcal{E}$ with ground-truth change points $c(E)$, and detected change points $d_p(E)$, we define the assignments of detected to ground-truth change points, $assign(d_p(E), c(E), lag) \subseteq d_p(E) \times c(E)$, as those assignments induced by the linear program:

$$
\begin{aligned}
\max \quad & \sum_{d \in d_p(E)} \sum_{c \in c(E)} x_{d,c} \\
\min \quad & \sum_{d \in d_p(E)} \sum_{c \in c(E)} x_{d,c} \cdot (d-c)^2 \\
\text{s.t.} \quad & \sum_{d \in d_p(E)} x_{d,c} \leq 1 && \forall c \in c(E) \qquad (1) \\
& \sum_{c \in c(E)} x_{d,c} \leq 1 && \forall d \in d_p(E) \qquad (2) \\
& \sum_{d \in d_p(E)} \sum_{c \in c(E)} x_{d,c} \cdot |d-c| \leq lag && \forall d \in d_p(E) \qquad (3) \\
& x_{d,c} \in \{0,1\} && \forall d \in d_p(E), c \in c(E)
\end{aligned}
$$

The linear program assigns at most one detected concept drift to a ground-truth drift and vice versa. In some use cases, it might be

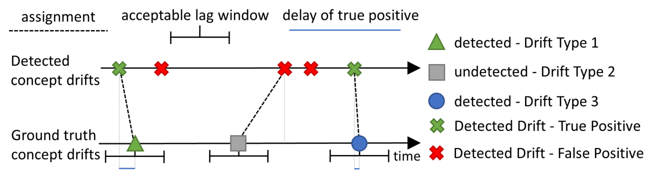


Figure 1: Different artifacts involved in calculating our evaluation metrics.

acceptable if a PCD detection technique generates multiple detected concept drifts for the same ground-truth drift. Therefore, we, also, provide a sensitivity analysis of constraint (1) in our evaluation. We evaluate the relative change of evaluation results to quantify how much the bipartite matching impacts the results.

5.4 Accuracy

Evaluation Techniques. Accuracy is measured through precision and recall using true and false positives of a detection. All evaluations agree on this process, individual techniques only differ in determining true and false positives. Similar to our framework, several approaches define a lag window assigning true positives [29, 35, 72]. Other approaches do not explicitly state that they use a lag window, but implicitly do so [8, 9, 12, 22, 24, 35, 39, 41, 50, 57, 70, 71]. Furthermore, one approach uses an infinite lag window, essentially counting the number of detected drifts and comparing it to the number of actual drifts [59]. However, this is a skewed measure as it does not factor in actual timely proximity [48]. One technique does not classify concept drifts themselves as true or false positives, but events. These are assigned to distributions given by the detected change points, using change points as indications that the detected distribution changes [30]. This approach does not scale beyond two alternating distributions since the assignment would be ambiguous. Other approaches just perform a qualitative discussion of results, not explicitly quantifying their accuracy [1, 23, 61].

Framework's Evaluation. The bipartite matching employed by our framework already yields true and false positives of the detection: If a detected change point is assigned to a ground-truth change point, it is a true positive. If it is not assigned, it is a false positive. Based on these assignments, we calculate precision and recall and quantify the accuracy with the F1-score. We employ the same technique as [29, 38, 72]. We retrieve multiple values to quantify accuracy, one for each parameter setting of the parameter sample. We use the maximal retrieved accuracy simulating the best-case scenario of optimal parameter setting. Note that the sensitivity to parameters is evaluated through another evaluation goal.

Definition 4 (Accuracy). *Let $E \subseteq \mathcal{E}$ be an event log, $p \in P \subseteq \mathcal{P}$ a parameter of the parameter sample set P and d be a PCD detection technique. Let $lag \in \mathbb{N}$ be an acceptable lag window. The precision of the detection is $prec_E(d_p) = \frac{|assign(d_p(E),c(E),lag)|}{|d_p(E)|}$, the recall is $recall_E(d_p) = \frac{|assign(d_p(E),c(E),lag)|}{|c(E)|}$. The accuracy of the detection is quantified by $F1_E(d_p) = 2 \cdot \frac{prec_E(d_p) \cdot recall_E(d_p)}{prec_E(d_p)+recall_E(d_p)}$. The accuracy of the PCD detection technique is the maximal F1-score over the parameter set, i.e., $acc_E(d,P) = \max_{p \in P} F1_E(d_p)$.*

5.5 Latency

Evaluation Techniques. Latency is, generally, quantified as the distance between the detected concept drift and the ground-truth concept drift. In the literature, there are two main approaches to quantifying the distance between detected and actual drift. First, the distance is defined as the number of process executions occurring between the change points of detected and actual change drift [34, 35, 42]. While this seems counter-intuitive, this metric can accurately quantify how many end-to-end runs through the process would be incorrectly classified. Second, the distance is defined as the number of timesteps that lie between the detected and ground-truth drift [46, 50, 72]. Therefore, it provides an average time lag for a detection. These two measures are different quantifications of the same distance. However, the first one provides more insights into the information needs of an algorithm to detect a concept drift, i.e., if we do not adapt, how many process executions are impacted?

Furthermore, some papers do not explicitly state the employed technique to quantify latency [22, 30].

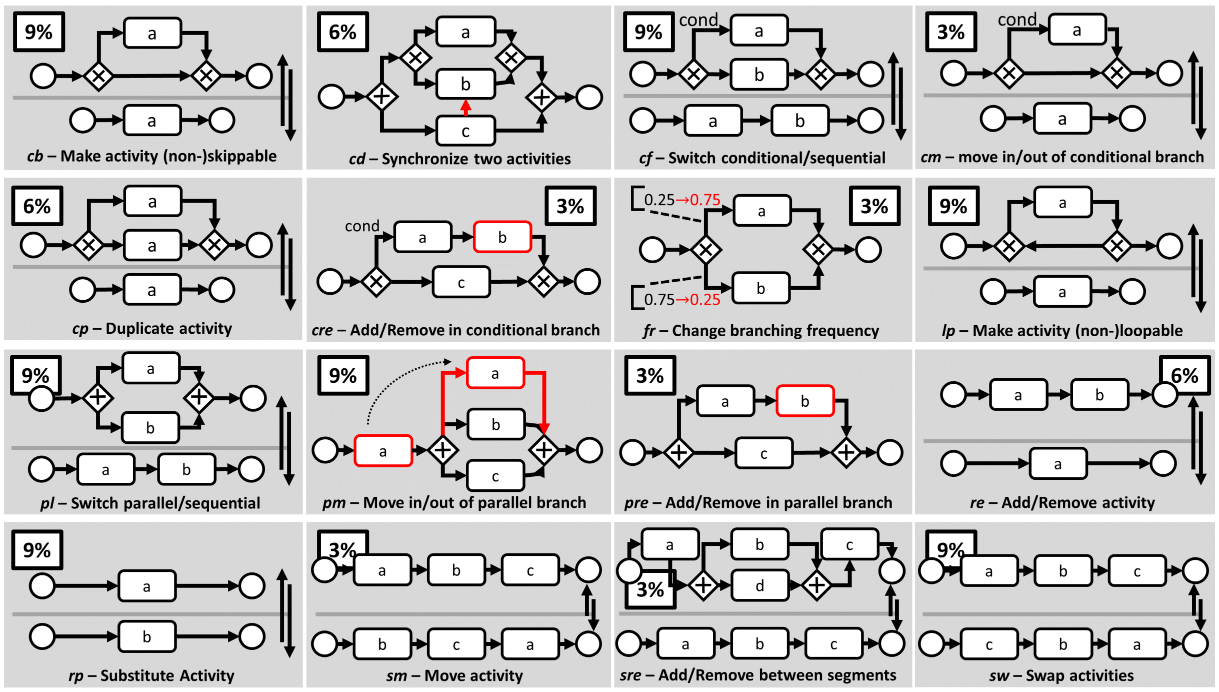
Framework's Evaluation. The employed bipartite matching provides assignments of detected to ground-truth concept drifts. Based on the assigned pairs, we calculate the number of process executions starting between the drifts' timesteps and quantify the detection latency. This is similar to the latency evaluation technique employed by [34, 35, 42]. Similar to accuracy, we choose the best value among the possible parameter settings to provide the best-case scenario.

Definition 5 (Latency). *Let $E \subseteq \mathcal{E}$ be an event log, $p \in P \subseteq \mathcal{P}$ a parameter of the parameter sample set P and d be a PCD detection technique. Let $lag \in \mathbb{N}$ be an acceptable lag window. For a timestamp $t \in \{\pi_{time}(e) \mid e \in E\}$, the number of cases starting before this timestamp is given by $nex(t) = |\{\langle e_1, \ldots, e_n \rangle \in \pi_{cases} \mid \pi_{time}(e_1) < t\}|$. The latency of a detection is defined by $lat_E(d_p) = \frac{1}{|assign(d_p(E),c(E),lag)|} \sum_{(x,y) \in assign(d_p(E),c(E),lag))} |nex(x) - nex(y)|$. The latency of the PCD detection technique is the minimal latency over the parameter set $lat_E(d,P) = \min_{p \in P} lat_E(d_p)$. The relative latency w.r.t. the acceptable lag window is defined by $rlat_E(d,P) = 1 - \frac{lat_E(d,P)}{lag}$.*

There is a trade-off between latency and accuracy depending on the *lag* window. A low *lag* value will exclude change points with higher latency, lowering accuracy but improving latency. A high *lag* value will include change points with higher latency, improving accuracy but worsening latency. Therefore, approaches with higher latency will achieve worse results w.r.t. the combined measures of accuracy and latency, independent of the *lag* value.

5.6 Versatility

Evaluation Techniques. Versatility is evaluated by comparing the performance of a PCD detection technique on drifts of different change types. Many algorithms are evaluated using a comprehensive set of event logs generated through systematically applying process change patterns to a process [1, 13, 15, 29, 30, 34, 35, 42, 50, 59, 70–72]. Some approaches use a more restrictive technique, comparing the performance of the PCD detection technique only for a subset of change patterns [8, 9, 12, 33, 38] (not explicitly targeting versatility). The omission of some change patterns will arguably worsen the reliability of versatility results since important, differentiating patterns between approaches might be left out.



9%
a
a
cb – Make activity (non-)skippable
6%
a
b
c
cd – Synchronize two activities
9%
cond
a
b
a
b
cf – Switch conditional/sequential
3%
cond
a
a
cm – move in/out of conditional branch
6%
a
a
a
cp – Duplicate activity
cond
a
b
c
3%
cre – Add/Remove in conditional branch
0.25→0.75
a
b
0.75→0.25
3%
fr – Change branching frequency
9%
a
a
lp – Make activity (non-)loopable
9%
a
b
a
b
pl – Switch parallel/sequential
9%
a
a
b
c
pm – Move in/out of parallel branch
3%
a
b
c
pre – Add/Remove in parallel branch
a
b
6%
a
re – Add/Remove activity
9%
a
b
rp – Substitute Activity
3%
a
b
c
b
c
a
sm – Move activity
a
b
c
3%
d
a
b
c
sre – Add/Remove between segments
9%
a
b
c
c
b
a
sw – Swap activities

Figure 2: Different drift types contained in the collected data sets (BPMN diagrams) [67].

Framework's Evaluation. The employed event logs all contain ground-truth concept drifts (cf. Table 3). Two sets of event logs [15, 42] are created through the explicit application of change patterns [67], i.e., also their underlying change type is known. We use these two event log sets to evaluate versatility. We determine which and how many patterns can be detected by the PCD detection technique, i.e., the recall given a concept drift type. We average over all concept drift type's scores to retrieve one number quantifying versatility. To drill down on this measure, we show the individual scores across the drift types providing insights into potential weaknesses. Like accuracy and latency, we choose the best value among the parameter settings to show the best-case scenario.

Definition 6 (Versatility). *Let $E \subseteq \mathcal{E}$ be an event log, $p \in P \subseteq \mathcal{P}$ a parameter of the parameter sample set P, d be a PCD detection technique, and lag $\in \mathbb{N}$ be an acceptable lag window. Let $T = \{1, \ldots, n\}$ with $n \in \mathbb{N}$ be a set of concept drift types. $c_i(E) \subseteq c(E)$ for $i \in T$ such that $c_j(E) \cap c_k(E) = \emptyset \; \forall j, k \in T, i \neq j$ are the sets of concept drifts of type i. $a'_i(d_p, E) = \{(d, c) \in assign(d_p(E), c(E), lag) \mid c \in c_i(E)\}$ is the subset of assigned concept drifts of type $i \in T$. $vers_E(d_p) = \frac{1}{|T|} \sum_{i \in T} \frac{|a'_i(d_p, E)|}{|c_i|}$ is the versatility of the detection. The versatility of the PCD detection technique is the maximal versatility over the parameter set $vers_E(d, P) = \max_{p \in P} vers_E(d_p)$.*

The versatility is effectively quantified as the average recall for each concept drift type. We can not incorporate precision since this would require algorithms to provide the type of concept drift[1]. As, in general, this information is not provided by PCD detection techniques, we have to stick to recall. However, recall already provides shows which drift types can be detected by a technique.

[1]Some algorithms providing also the drift type have been proposed in the last years. The term concept drift characterization summarizes these techniques. However, the problem of concept drift detection itself does not require the characterization of the drift. Therefore, we do not evaluate for accurate characterization.

5.7 Scalability

Evaluation Techniques. The techniques evaluating a PCD detection technique's scalability can be grouped into two categories: evaluation against the algorithm itself and evaluation against other algorithms. When evaluating the algorithm itself, authors use one of three techniques: First, they provide the complexity of the algorithm [22, 46, 71]. Second, they depict computation time statistics on certain event logs and conclude the scalability of the algorithm [13, 34, 35, 42]. Third, they investigate the evolution of computation times among differing parameter settings such as input size or window size [38]. When evaluating an algorithm against other algorithms, researchers use a set of event logs and compare the computation times [15, 24, 29, 30, 72].

Framework's Evaluation. We evaluate for scalability by comparing the computation times per event. We do not compare complexities for several reasons: First, PCD detection techniques often combine many different sub-techniques whose complexity is dependent on the chosen implementation. Second, many algorithms neither provide their complexity nor any pseudocode from which the complexity could be derived. Third, we are interested in the applicability and, therefore, the scalability on typical event logs. A complexity analysis would yield limited insights into the actual applicability. We calculate the average computation time per event across all event logs of our evaluation framework (cf. Table 3) to provide realistic scalability metrics on typical event logs.

5.8 Parameter Sensitivity

Evaluation Techniques. Evaluating parameter sensitivity across the whole parameter space is often not possible (cf. Subsection 5.2). In the related work, the parameter sensitivity is evaluated either with a quantitative or qualitative approach: On the quantitative side, parameter sensitivity is specified through an equidistant sampling of parameter values and specifying the impact on accuracy [24, 29, 72] or accuracy and latency combined [34, 35, 42]. On the qualitative side, a discussion about the impact of different parameter values on the algorithm's results is provided [1, 39].

Framework's Evaluation. All quantitative evaluations in the introduced PCD detection techniques agree on the evaluation technique for parameter sensitivity: sampling an equidistant set of parameters and employing the evaluation techniques used to quantify the quality of the algorithm before. Therefore, we use the parameter sample discussed Subsection 5.2 and calculate the average performance. We calculate the performance given a parameter setting by determining the harmonic mean of accuracy, latency, and versatility. We determine the interquartile range, i.e., the difference between the first and third quartile of the set of values, to quantify the spread in the result's quality in one single metric.

Definition 7. *Let $E \subseteq \mathcal{E}$ be an event log, $p \in P \subseteq \mathcal{P}$ a parameter of the parameter sample set P and d be a PCD detection technique. Let $lag \in \mathbb{N}$ be an acceptable lag window. $IQR(X)$ denotes the interquartile range of a set $X \subseteq \mathbb{R}$. The parameter sensitivity is quantified through the interquartile range of the harmonic mean of accuracy, latency, and versatility over the parameter space $sens_E(d, P) = IQR(\{\frac{3acc_E(d_p)\cdot rlat_E(d_p)\cdot vers_E(d_p)}{acc_E(d_p)\cdot rlat_E(d_p)+acc_E(d_p)\cdot vers_E(d_p)+rlat_E(d_p)\cdot vers_E(d_p)} \mid p \in P\})$.*

5.9 Robustness

Evaluation Techniques. Noise is often present in real-life data. Robustness describes the PCD detection technique's ability to handle such noise and, therefore, real-life event data. There are two main evaluation techniques for robustness: Qualitatively assessing the results achieved on real-life data sets and quantifying the impact of artificially introduced noise on the detection results. Several techniques are evaluated using a real-life event log and discussing the detection results [9, 22–24, 29, 34, 35, 38, 42, 46, 61, 71]. Two PCD detection techniques are evaluated by using event logs with increasing noise levels, either removing events of randomly chosen process executions [42] or by adding random, non-fitting events to process executions [15]. The evaluation of Ostovar et al. [40] is also notable: Even though the authors evaluate for process drift characterization instead of process drift detection they evaluate with artificially generated event logs of varying noise levels. One evaluation provides a discussion about the ability to handle noise between different tools [39].

Framework's Evaluation. To quantify the robustness of an approach, we use two sets of event logs with artificially introduced noise [15, 40]. One set defines noise as the absence of correct information while the other set defines noise as the presence of incorrect information. Analogously to parameter sensitivity, we determine the average harmonic mean of accuracy, latency, and versatility among event logs of different noise levels. We quantify the robustness as the relative decline per percent of noise. The performance for the noiseless log is taken as a baseline. This is equivalent to comparing areas under the curve for the baseline performance and the performance evolution under increasing noise. If an approach's performance increases with noise, the robustness value might be larger than 1. We use the maximum value achieved in the parameter set to present the best case.

Table 4: Evaluation results. The relative changes when allowing multiple detected drifts per ground-truth drift are depicted in parentheses.

Algorithm	Accuracy	Lat.	Vers.	Scalability	P. Sensitivity	Robustness
ADWIN J	0.75 (→)	66.72	0.5938	0.0044s	0.0482 (→)	0.9531(→)
EMD	0.8333 (→)	43.9	**0.875**	0.0157s	**0.017** (↓ 40%)	1.0026(→)
J-MEASURE	0.7901 (→)	52.63	0.7708	0.0111s	0.1233 (↓ 1%)	1.05 (↑ 1%)
LCDD	0.6742 (↑ 2%)	26.0	0.7083	**2.4593e-6s**	0.2317 (↑ 13%)	0.9908 (↓ 3%)
PGM	0.7647 (→)	48.45	0.6563	4.977e-4s	0.0635 (→)	**1.0465** (→)
PRODRIFT	0.5965 (→)	**18.29**	0.3854	0.0042s	0.0198 (→)	0.6 (→)
RINV	**0.8642** (→)	54.04	0.8542	5.8217e-5s	0.0747 (↑ 17%)	0.6 (→)
WINDOW C.	0.7429 (→)	41.57	0.625	0.0111s	0.1092 (→)	1.0287 (→)

Definition 8 (Robustness). *Let $E_0 \subseteq \mathcal{E}$ be a noiseless event log. $noi : \mathbb{N} \times \mathcal{P}(\mathcal{E}) \to \mathcal{P}(\mathcal{E})$ induces noise of a certain percentage to an event log. Let $N \subseteq \mathbb{N}$ be noise of varying percentage levels. Let $p \in P \subseteq \mathcal{P}$ be a parameter of the parameter sample set P and d be a PCD detection technique. $sort(X) = \{(i, j) \in X{\times}X \mid i{<}j \land \neg\exists_{k\in X} i{<}k{<}j\}$ provides a set of ascending tuples from $X \in \mathbb{N}$. The performance of a detection technique for an event log E and a parameter setting p is defined as $per_E(d_p) = \frac{3acc_E(d_p)\cdot rlat_E(d_p)\cdot vers_E(d_p)}{acc_E(d_p)\cdot rlat_E(d_p)+acc_E(d_p)\cdot vers_E(d_p)+rlat_E(d_p)\cdot vers_E(d_p)}$ The baseline performance is defined as $base_{E_0,N}(d_p) = per_{E_0}(d_p) \cdot \max_{n\in N} n$. The relative performance is defined as $relper_{E_0,N}(d_p) = \sum_{(i,j)\in sort(N)} \frac{1}{2}(per_{noi(E_0,i)}(d_p) + per_{noi(E_0,j)}(d_p)) \cdot (j - i)$. The robustness given a parameter setting p of PCD detection technique d is defined as $rob_E(d_p) = \frac{relper_{E_0,N}(d_p)}{base_{E_0,N}(d_p)}$. The robustness of the PCD detection technique is the maximum robustness over the whole parameter set $rob_E(d, P) = \max_{p\in P} rob_E(d_p)$.*

6 EXPERIMENTAL EVALUATION

In this section, we present the results of applying our defined evaluation techniques on the selected PCD detection techniques using the set of available event logs. In addition to the overall scores of algorithms for each evaluation goal, we provide an in-depth visualization of the results for each evaluation goal. The overall results are depicted in Table 4.

Experimental Setting. We apply each of the selected PCD detection techniques of Section 4 to the event logs introduced in Table 3 using varying parameter settings as described in Subsection 5.2. We implemented the PCD detection techniques ADWIN J & ADWIN WC [38], EMD [12], J-MEASURE [8, 9], LCDD [29], PGM [50], PRODRIFT [34, 35], RINV [72], and WINDOW COUNT [8, 9]. The event log sets adopted from [8, 15, 42] contain in total 90 event logs. The parameter sampling strategy yields a total of 23670 algorithm runs. The presented results are based on a lag window size of the equivalent time of 200 process execution starts. The evaluation framework along with the implemented algorithms is available on GitHub[2]. The experiments ran for roughly 3 days on our employed

[2]https://github.com/cpitsch/cdrift-evaluation

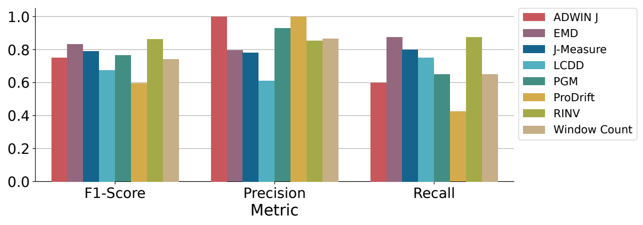


Figure 3: Precision, recall, and F1-score of algorithms averaged over noiseless event logs and parameter settings.

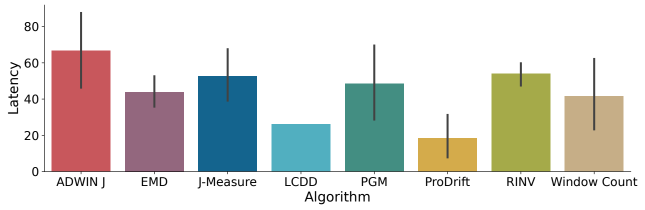


Figure 4: The average delay und stadard deviation of a detected concept drift w.r.t. its actual location.

machine[3] using multiprocessing on 22 cores. The pre-computed results of applying all techniques to the event logs are available in tabular format in the GitHub repository. Using the provided script, results for different lag window sizes can be computed and visualized. In our experiments, ADWIN WC was not able to detect any of the ground-truth concept drifts due to high p-values ($p > 0.9$) when performing hypothesis testing. Since these values are way off detecting any significant difference (usually $p < 0.05$) between distributions, we exclude the approach from our results.

6.1 Accuracy

The accuracy results are depicted in Figure 3. The best accuracy score is achieved by RINV, closely followed by EMD. The accuracy is calculated as F1-score of precision and fitness. Therefore, we can further analyze the F1-score by drilling down accuracy to precision and recall. Some algorithms like ADWIN J or PRODRIFT have a high precision and low recall. Therefore, they detected comparably few concept drifts, but the ones detected are actual ground-truth concept drifts. In a setting where there is a very low tolerance for false positives, these algorithms might be a reasonable choice. When the main objective is catching as many ground-truth drifts as possible, reasonable choices would be EMD or RINV.

One surprising observation is the accuracy of ADWIN J compared to J-MEASURE: Since one is the application of adaptive window size to the other algorithm, one would expect an improvement in accuracy. This is, however, not the case. A potential reason might be an overshooting window size. When there is a stable behavior, the window size grows and drifts might be detected with a large offset.

6.2 Latency

The latency results are depicted in Figure 4. The bar chart indicates the average delay for each detected concept drift of an algorithm. The delay is measured as the number of process executions that lie in between the ground truth and the detected drift, i.e., that are incorrectly classified. The lower the delay the better the algorithm locates the change point. While some algorithms like ADWIN J exhibit high delays with high variation, especially PRODRIFT stands out with low delays. The likely reasons for PRODRIFTS superior performance is the custom adaptive windowing strategy. By assessing the statistical variation of runs within the windows, PRODRIFT changes the window size. This custom adaptive windowing performs much better than ADWIN for ADWIN J, which exhibits the worst latency performance. In general, applying PRODRIFTS adaptive windowing might significantly increase other approaches' performances. However, given the limited accuracy of PRODRIFT, one should also examine the effects on other metrics.

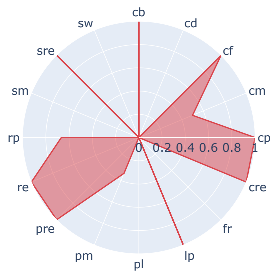
[3]AMD Ryzen Threadripper 1920X, 126 GB RAM

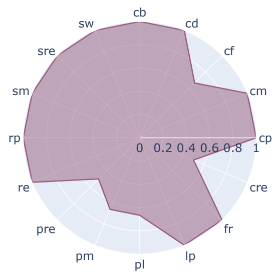
6.3 Versatility

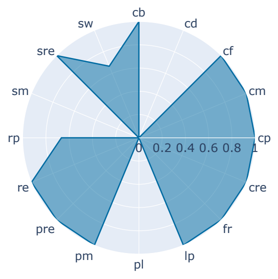
We depict the results of our versatility evaluation in Figure 5. The visualization depicts a radar chart with the recall for different process change patterns [67] for each algorithm. Each change pattern is abbreviated with commonly used acronyms [41] illustrated in Figure 2. The visualization offers a drill down into the different algorithm's abilities to detect different change patterns. While especially EMD is able to detect at least some concept drifts induced by every change pattern, algorithms like PRODRIFT, ADWIN J, and WINDOW COUNT are only able to detect certain subsets of change patterns. Furthermore, no algorithm is perfectly able to detect changes from parallel to sequential relationships between activities (*pl*). The most severe systematic issue can be observed for *cd* (synchronizing two parallel branches): Only EMD is able to capture any drift of such a type. EMDS superior versatility is linked to its analysis of the variant distribution: Since variants describe all the end-to-end control-flow behavior captured in the event logs, even long-term dependency changes will be captured using variant analysis. Many of the approaches only use activity pair features (e.g. ADWIN J) or build only on directly-follows relationships (α-relations of PRODRIFT) that cannot capture complex long-term relationships. As EMD is the only technique fully employing variants (cf. Table 2) and also the only technique that shows promising versatility, improvements in versatility can likely be made by moving from pair-wise activity features and features derived from directly-follows relationships towards variant based features. Furthermore, some features cannot detect frequency-related changes (*fr*) as they only incorporate binary information about relationships, e.g., the directly-follows relationships extracted by RINV and LCDD.

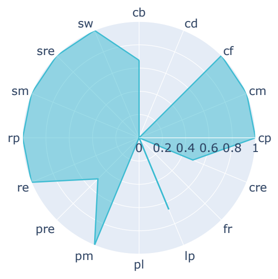
6.4 Scalability

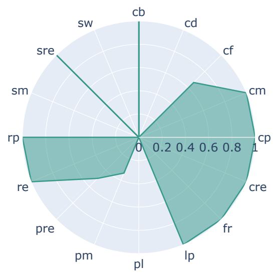
The average computation time per event is depicted in Figure 6. LCDD exhibits by far the best scalability with processing times per event being only a few microseconds. RINV, the second technique from the MV Feature Vector Series class, also shows promising scalability, ranging within a few dozen microseconds per event. Both techniques only operate on directly-follow relationships that can be computed with a single pass over the data. However, as seen in the versatility section, employing directly-follows relationships

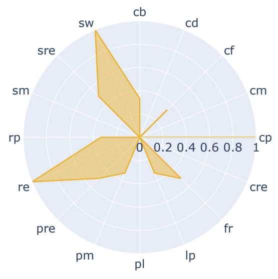

(a) ADWIN J

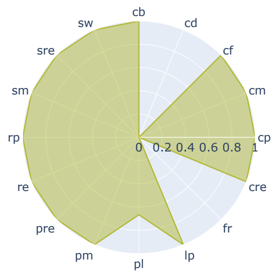

(b) EMD


(c) J-Measure


(d) LCDD


(e) PGM


(f) ProDrift


(g) RINV

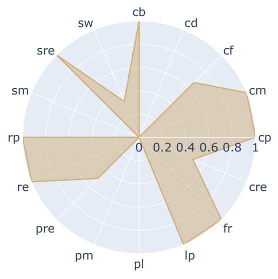

(h) Window Count

Figure 5: Recall of PCD detection techniques for different change patterns. The change patterns are illustrated in Figure 2.

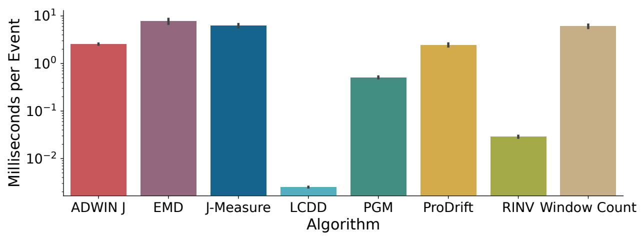


Figure 6: Boxplots of the computation times collected for all logs and all parameter settings for each algorithm.

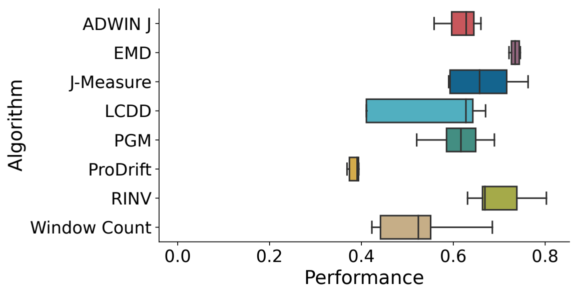


Figure 7: Performance (harmonic mean of accuracy, latency, and versatility) across all event logs and parameter settings.

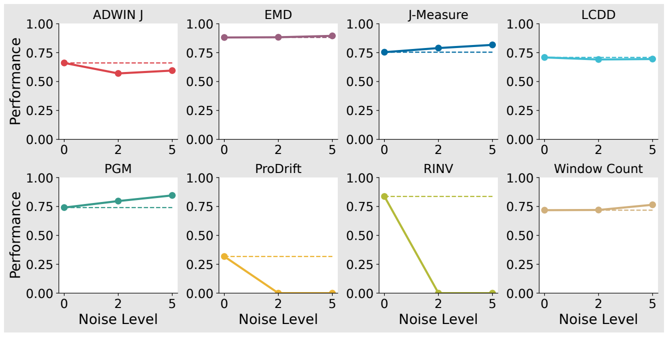
limits the versatility of a technique as some change patterns become undetectable. There is a trade-off between versatility and scalability considering EMDs long computation times. A new technique aiming to strike both metrics could propose efficient variant incorporation.

6.5 Parameter Sensitivity

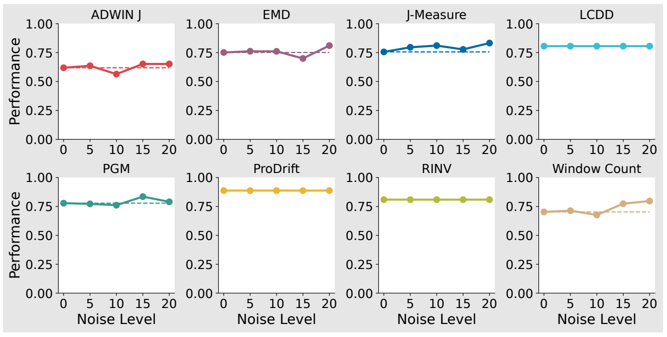
Figure 7 depicts boxplots of the achieved harmonic mean of accuracy, latency, and versatility across a parameter set for each algorithm. Algorithms not sensitive to the parameter choice should exhibit low levels of variance. An ideal algorithm would score high values of accuracy, latency, and versatility across all parameters. While we can observe significant differences in parameter sensitivity, few overall trends for classes of algorithms are deductible. One commonality is the superior parameter sensitivity of algorithms with few parameters, especially if these parameters are window and step size. EMD and PRODRIFT, both requiring only window and step size, show little variance in their performance. LCDD on the contrary, which requires two window sizes and a stable period, shows a large variance across parameter settings. Many parameters increase the number of possible parameter settings (cartesian product) and can, therefore, lead to disadvantageous combinations. A technique with few parameters seems to be preferable.

6.6 Robustness

We present the evolution of the performance (harmonic mean of accuracy, latency, and versatility) for each algorithm and event log with differing noise levels in Figure 8. The baseline, i.e., the performance without noise, is indicated as a dashed line. We immediately observe one interesting phenomenon: The performance of many algorithms increases with the presence of noise whereas one would intuitively expect a decrease. This might be explained through the immanent assumption under which PCD detection techniques are developed: The algorithms should work in a real-life, noisy environment. Therefore, researchers have chosen techniques that factor in some kind of noise tolerance, e.g., hypothesis tests, earth mover's



(a) First set of logs with noise as additional, non-fitting events [42].



(b) Second set of logs with noise as absence of fitting events [15].

Figure 8: The performance (harmonic mean of accuracy, latency, and versatility) under induced noise of different levels.

distance, or model discovery with integrated noise handling. As we compare the areas below the curves, a robustness measure of more than 1 indicates an increased performance with noise.

The two sets of logs are generated with two orthogonal definitions of noise: The first set of logs with noise as additional, non-fitting events and the second set of logs with noise as the absence of fitting events. The first type of noise seems to be more consequential for many approaches, especially for RINV. For RINV, these non-fitting events disrupt the stable periods, leading to incorrect classifications. The technique could benefit from a frequency-based stable period that allows for some noise. Another possible improvement might be the preprocessing of event data to generously delete process executions with potentially faulty introduced events.

6.7 Overall Evaluation

We discuss the overall evaluation results depicted in Table 4 by, first, investigating the effect of the bipartite matching from detected concept drift to ground-truth concept drifts and, second, summarizing the overall results. Table 4 displays the relative changes when dropping the requirement that a ground-truth drift can only be assigned to one detected drift. The results mostly stay the same. Only small changes in the accuracy can be observed for LCDD. EMD significantly improves its already good sensitivity. It probably contains several double detections for a parameter setting. In total, the employed bipartite matching does not skew the results.

According to our evaluation, there is no PCD detection technique that dominates all other techniques. However, our evaluation points to several aspects that can strengthen a technique for different evaluation goals. A suitable adaptive windowing (cf. PRODRIFT) can reduce latency while incorporating variant analysis makes all change patterns detectable (cf. EMD). While activity pair/directly-follows-based features are linked to reduced versatility in terms of frequency patterns and are susceptible to noise, they can enable very fast computation (cf RINV, LCDD). The most promising direction for further development of PCD detection techniques is variant-based adaptations of MV Feature Vector Series techniques (RINV,LCDD) and efficient EMD adaptations. When considering the relationship between the taxonomy of [32] and ours, i.e., the absence of multiple hypothesis testing methods, another promising direction is the development of ensemble techniques that cover different change patterns, enabled by our evaluation.

7 CONCLUSION

This paper presented a general framework for the experimental evaluation of PCD detection techniques. We collected common PCD evaluation goals and summarized them in a framework with one evaluation technique for each of these goals. We presented the field of process concept drift detection and selected a representative sample of publicly available detection techniques. The application of our framework to PCD detection techniques maps out strengths and weaknesses of different approaches and guides researchers in developing and practitioners in choosing PCD detection techniques. Furthermore, the framework can be extended with additional PCD detection techniques and event logs. We provide several implications for the development of further techniques: Adaptive windowing should be used to reduce latency, features should be calculated based on variants to capture all control-flow change patterns, and features should incorporate frequencies to cover frequency-based change patterns and handle noise. Furthermore, multiple hypothesis test detection techniques (e.g. ensemble classifiers) are a promising direction for further development. As for practical implications, our evaluation has shown that several techniques can cost-efficiently be deployed in practice, with low computation times and good results. The techniques could be implemented into model-learning pipelines to notify a necessary model update without significant cost increases, such as process discovery pipelines or predictive process monitoring pipelines.

A future line of research should address real-life event logs with ground-truth concept drifts. For example, through extensive surveys of process stakeholders or a broader socio-economic analysis, a real-life event log could be annotated with a large set of ground-truth event logs.

ACKNOWLEDGMENTS

We thank the Alexander von Humboldt (AvH) Stiftung for supporting our research. Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy—EXC-2023 Internet of Production—390621612.

REFERENCES

[1] Rafael Accorsi and Thomas Stocker. 2011. Discovering Workflow Changes with Time-Based Trace Clustering. In *Data-Driven Process Discovery and Analysis - First International Symposium, SIMPDA 2011, Campione d'Italia, Italy, June 29 - July 1, 2011, Revised Selected Papers*, Karl Aberer, Ernesto Damiani, and Tharam S. Dillon (Eds.). Springer, 154–168. https://doi.org/10.1007/978-3-642-34044-4_9

[2] Jan Niklas Adams, Sebastiaan J. van Zelst, Thomas Rose, and Wil M. P. van der Aalst. 2023. Explainable concept drift in process mining. *Inf. Syst.* 114 (2023), 102177. https://doi.org/10.1016/j.is.2023.102177

[3] Cesare Alippi, Giacomo Boracchi, and Manuel Roveri. 2017. Hierarchical Change-Detection Tests. *IEEE Trans. Neural Networks Learn. Syst.* 28, 2 (2017), 246–258. https://doi.org/10.1109/TNNLS.2015.2512714

[4] Cesare Alippi and Manuel Roveri. 2008. Just-in-Time Adaptive Classifiers - Part I: Detecting Nonstationary Changes. *IEEE Trans. Neural Networks* 19, 7 (2008), 1145–1153. https://doi.org/10.1109/TNN.2008.2000082

[5] Adriano Augusto, Raffaele Conforti, Marlon Dumas, Marcello La Rosa, and Artem Polyvyanyy. 2019. Split miner: automated discovery of accurate and simple business process models from event logs. *Knowledge and Information Systems* 59, 2 (2019), 251–284. https://doi.org/10.1007/s10115-018-1214-x

[6] Albert Bifet and Ricard Gavaldà. 2007. Learning from Time-Changing Data with Adaptive Windowing. In *Proceedings of the Seventh SIAM International Conference on Data Mining, April 26-28, 2007, Minneapolis, Minnesota, USA*. SIAM, 443–448. https://doi.org/10.1137/1.9781611972771.42

[7] Isvani Inocencio Frías Blanco, José del Campo-Ávila, Gonzalo Ramos-Jiménez, Rafael Morales Bueno, Agustín Alejandro Ortiz Díaz, and Yailé Caballero Mota. 2015. Online and Non-Parametric Drift Detection Methods Based on Hoeffding's Bounds. *IEEE Trans. Knowl. Data Eng.* 27, 3 (2015), 810–823. https://doi.org/10.1109/TKDE.2014.2345382

[8] R. P. Jagadeesh Chandra Bose, Wil M. P. van der Aalst, Indre Zliobaite, and Mykola Pechenizkiy. 2011. Handling Concept Drift in Process Mining. In *Advanced Information Systems Engineering - 23rd International Conference, CAiSE 2011, London, UK, June 20-24, 2011. Proceedings*, Haralambos Mouratidis and Colette Rolland (Eds.). Springer, 391–405. https://doi.org/10.1007/978-3-642-21640-4_30

[9] R. P. Jagadeesh Chandra Bose, Wil M. P. van der Aalst, Indre Zliobaite, and Mykola Pechenizkiy. 2014. Dealing With Concept Drifts in Process Mining. *IEEE Trans. Neural Networks Learn. Syst.* 25, 1 (2014), 154–171. https://doi.org/10.1109/TNNLS.2013.2278313

[10] Melike Bozkaya, Joost Gabriels, and Jan Martijn van der Werf. 2009. Process diagnostics: a method based on process mining. In *2009 International Conference on Information, Process, and Knowledge Management*. IEEE, 22–27. https://doi.org/10.1109/eKNOW.2009.29

[11] Dominic Breuker, Martin Matzner, Patrick Delfmann, and Jörg Becker. 2016. Comprehensible Predictive Models for Business Processes. *MIS Quarterly* 40, 4 (2016), 1009–1034. https://www.jstor.org/stable/26629686

[12] Tobias Brockhoff, Merih Seran Uysal, and Wil M. P. van der Aalst. 2020. Time-aware Concept Drift Detection Using the Earth Mover's Distance. In *2nd International Conference on Process Mining, ICPM 2020, Padua, Italy, October 4-9, 2020*, Boudewijn F. van Dongen, Marco Montali, and Moe Thandar Wynn (Eds.). IEEE, 33–40. https://doi.org/10.1109/ICPM49681.2020.00016

[13] Josep Carmona and Ricard Gavaldà. 2012. Online Techniques for Dealing with Concept Drift in Process Mining. In *Advances in Intelligent Data Analysis XI - 11th International Symposium, IDA 2012, Helsinki, Finland, October 25-27, 2012. Proceedings*, Jaakko Hollmén, Frank Klawonn, and Allan Tucker (Eds.). Springer, 90–102. https://doi.org/10.1007/978-3-642-34156-4_10

[14] Josep Carmona, Boudewijn F. van Dongen, Andreas Solti, and Matthias Weidlich. 2018. *Conformance Checking - Relating Processes and Models*. Springer. https://doi.org/10.1007/978-3-319-99414-7

[15] Paolo Ceravolo, Gabriel Marques Tavares, Sylvio Barbon Junior, and Ernesto Damiani. 2020. Evaluation Goals for Online Process Mining: a Concept Drift Perspective. *IEEE Transactions on Services Computing* (2020). https://doi.org/10.1109/TSC.2020.3004532

[16] Alfonso E. Márquez Chamorro, Isabel A. Nepomuceno-Chamorro, Manuel Resinas, and Antonio Ruiz-Cortés. 2022. Updating Prediction Models for Predictive Process Monitoring. In *Advanced Information Systems Engineering - 34th International Conference, CAiSE 2022, Leuven, Belgium, June 6-10, 2022, Proceedings*, Xavier Franch, Geert Poels, Frederik Gailly, and Monique Snoeck (Eds.). Springer, 304–318. https://doi.org/10.1007/978-3-031-07472-1_18

[17] Alfonso E. Márquez Chamorro, Isabel A. Nepomuceno-Chamorro, Manuel Resinas, and Antonio Ruiz-Cortés. 2022. Updating Prediction Models for Predictive Process Monitoring. In *Advanced Information Systems Engineering - 34th International Conference, CAiSE 2022, Leuven, Belgium, June 6-10, 2022, Proceedings*, Xavier Franch, Geert Poels, Frederik Gailly, and Monique Snoeck (Eds.). Springer, 304–318. https://doi.org/10.1007/978-3-031-07472-1_18

[18] Lei Du, Qinbao Song, Lei Zhu, and Xiaoyan Zhu. 2015. A Selective Detector Ensemble for Concept Drift Detection. *Comput. J.* 58, 3 (2015), 457–471. https://doi.org/10.1093/comjnl/bxu050

[19] Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A. Reijers. 2018. *Fundamentals of Business Process Management, Second Edition*. Springer. https://doi.org/10.1007/978-3-662-56509-4

[20] João Gama, Pedro Medas, Gladys Castillo, and Pedro Pereira Rodrigues. 2004. Learning with Drift Detection. In *Advances in Artificial Intelligence - SBIA 2004, 17th Brazilian Symposium on Artificial Intelligence, São Luis, Maranhão, Brazil, September 29 - October 1, 2004, Proceedings*, Ana L. C. Bazzan and Sofiane Labidi (Eds.). Springer, 286–295. https://doi.org/10.1007/978-3-540-28645-5_29

[21] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. 2014. A survey on concept drift adaptation. *ACM computing surveys* 46, 4 (2014), 1–37. https://doi.org/10.1145/2523813

[22] Marwan Hassani. 2019. Concept Drift Detection Of Event Streams Using An Adaptive Window. In *Proceedings of the 33rd International ECMS Conference on Modelling and Simulation, ECMS 2019 Caserta, Italy, June 11-14, 2019*, Mauro Iacono, Francesco Palmieri, Marco Gribaudo, and Massimo Ficco (Eds.). European Council for Modeling and Simulation, 230–239. https://doi.org/10.7148/2019-0230

[23] Bart Hompes, Joos C. A. M. Buijs, Wil M. P. van der Aalst, Prabhakar M. Dixit, and Hans Buurman. 2015. Detecting Change in Processes Using Comparative Trace Clustering. In *Proceedings of the 5th International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2015), Vienna, Austria, December 9-11, 2015 (CEUR Workshop Proceedings)*, Paolo Ceravolo and Stefanie Rinderle-Ma (Eds.), Vol. 1527. CEUR-WS.org, 95–108. http://ceur-ws.org/Vol-1527/paper7.pdf

[24] Angelo Impedovo, Paolo Mignone, Corrado Loglisci, and Michelangelo Ceci. 2020. Simultaneous Process Drift Detection and Characterization with Pattern-Based Change Detectors. In *Discovery Science - 23rd International Conference, DS 2020, Thessaloniki, Greece, October 19-21, 2020, Proceedings*, Annalisa Appice, Grigorios Tsoumakas, Yannis Manolopoulos, and Stan Matwin (Eds.). Springer, 451–467. https://doi.org/10.1007/978-3-030-61527-7_30

[25] Daniel Kifer, Shai Ben-David, and Johannes Gehrke. 2004. Detecting Change in Data Streams. In *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, VLDB 2004, Toronto, Canada, August 31 - September 3 2004*, Mario A. Nascimento, M. Tamer Özsu, Donald Kossmann, Renée J. Miller, José A. Blakeley, and K. Bernhard Schiefer (Eds.). Morgan Kaufmann, 180–191. https://doi.org/10.1016/B978-012088469-8.50019-X

[26] Wolfgang Kratsch, Jonas Manderscheid, Maximilian Röglinger, and Johannes Seyfried. 2021. Machine learning in business process monitoring: a comparison of deep learning and classical approaches used for outcome prediction. *Business & Information Systems Engineering* 63, 3 (2021), 261–276. https://doi.org/10.1007/s12599-020-00645-0

[27] Geetika T. Lakshmanan, Paul T. Keyser, and Songyun Duan. 2011. Detecting changes in a semi-structured business process through spectral graph analysis. In *2011 IEEE 27th International Conference on Data Engineering Workshops*. 255–260. https://doi.org/10.1109/ICDEW.2011.5767640

[28] Sander J. J. Leemans, Dirk Fahland, and Wil M. P. van der Aalst. 2013. Discovering Block-Structured Process Models from Event Logs - A Constructive Approach. In *Application and Theory of Petri Nets and Concurrency - 34th International Conference, PETRI NETS 2013, Milan, Italy, June 24-28, 2013. Proceedings*. Springer. https://doi.org/10.1007/978-3-642-38697-8_17

[29] Leilei Lin, Lijie Wen, Li Lin, Jisheng Pei, and Hedong Yang. 2020. LCDD: Detecting Business Process Drifts Based on Local Completeness. *IEEE Transactions on Services Computing* (2020), 1–1. https://doi.org/10.1109/TSC.2020.3032787

[30] Na Liu, Jiwei Huang, and Lizhen Cui. 2018. A Framework for Online Process Concept Drift Detection from Event Streams. In *2018 IEEE International Conference on Services Computing, SCC 2018, San Francisco, CA, USA, July 2-7, 2018*. IEEE, 105–112. https://doi.org/10.1109/SCC.2018.00021

[31] Rafael Lorenz, Julian Senoner, Wilfried Sihn, and Torbjørn H. Netland. 2021. Using process mining to improve productivity in make-to-stock manufacturing. *Int. J. Prod. Res.* 59, 16 (2021), 4869–4880. https://doi.org/10.1080/00207543.2021.1906460

[32] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, João Gama, and Guangquan Zhang. 2019. Learning under Concept Drift: A Review. *IEEE Trans. Knowl. Data Eng.* 31, 12 (2019), 2346–2363. https://doi.org/10.1109/TKDE.2018.2876857

[33] Daniela Luengo and Marcos Sepúlveda. 2011. Applying Clustering in Process Mining to Find Different Versions of a Business Process That Changes over Time. In *Business Process Management Workshops - BPM 2011 International Workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part I*, Florian Daniel, Kamel Barkaoui, and Schahram Dustdar (Eds.). Springer, 153–158. https://doi.org/10.1007/978-3-642-28108-2_15

[34] Abderrahmane Maaradji, Marlon Dumas, Marcello La Rosa, and Alireza Ostovar. 2015. Fast and Accurate Business Process Drift Detection. In *Business Process Management - 13th International Conference, BPM 2015, Innsbruck, Austria, August 31 - September 3, 2015, Proceedings*, Hamid Reza Motahari-Nezhad, Jan Recker, and Matthias Weidlich (Eds.). Springer, 406–422. https://doi.org/10.1007/978-3-319-23063-4_27

[35] Abderrahmane Maaradji, Marlon Dumas, Marcello La Rosa, and Alireza Ostovar. 2017. Detecting Sudden and Gradual Drifts in Business Processes from Execution

Traces. *IEEE Trans. Knowl. Data Eng.* 29, 10 (2017), 2140–2154. https://doi.org/10.1109/TKDE.2017.2720601

[36] Fabrizio Maria Maggi, Chiara Di Francescomarino, Marlon Dumas, and Chiara Ghidini. 2014. Predictive Monitoring of Business Processes. In *Advanced Information Systems Engineering - 26th International Conference, CAiSE 2014, Thessaloniki, Greece, June 16-20, 2014. Proceedings*, Matthias Jarke, John Mylopoulos, Christoph Quix, Colette Rolland, Yannis Manolopoulos, Haralambos Mouratidis, and Jennifer Horkoff (Eds.). Springer, 457–472. https://doi.org/10.1007/978-3-319-07881-6_31

[37] Marco Maisenbacher and Matthias Weidlich. 2017. Handling Concept Drift in Predictive Process Monitoring. In *2017 IEEE International Conference on Services Computing, SCC 2017, Honolulu, HI, USA, June 25-30, 2017*, Xiaoqing (Frank) Liu and Umesh Bellur (Eds.). IEEE Computer Society, 1–8. https://doi.org/10.1109/SCC.2017.10

[38] J. Martjushev, R. P. Jagadeesh Chandra Bose, and Wil M. P. van der Aalst. 2015. Change Point Detection and Dealing with Gradual and Multi-order Dynamics in Process Mining. In *Perspectives in Business Informatics Research - 14th International Conference, BIR 2015, Tartu, Estonia, August 26-28, 2015, Proceedings*, Raimundas Matulevicius and Marlon Dumas (Eds.). Springer, 161–178. https://doi.org/10.1007/978-3-319-21915-8_11

[39] Nicolas Jashchenko Omori, Gabriel Marques Tavares, Paolo Ceravolo, and Sylvio Barbon Jr. 2019. Comparing Concept Drift Detection with Process Mining Tools. In *Proceedings of the XV Brazilian Symposium on Information Systems, SBSI 2019, Aracaju, Brazil, May 20-24, 2019*, Fábio Gomes Rocha, Igor Vasconcelos, Rodrigo Pereira dos Santos, Davi Viana, and Scheila de Avila e Silva (Eds.). ACM, 31:1–31:8. https://doi.org/10.1145/3330204.3330240

[40] Alireza Ostovar, Sander J. J. Leemans, and Marcello La Rosa. 2020. Robust Drift Characterization from Event Streams of Business Processes. *ACM Trans. Knowl. Discov. Data* 14, 3 (2020), 30:1–30:57. https://doi.org/10.1145/3375398

[41] Alireza Ostovar, Abderrahmane Maaradji, Marcello La Rosa, and Arthur H. M. ter Hofstede. 2017. Characterizing Drift from Event Streams of Business Processes. In *Advanced Information Systems Engineering - 29th International Conference, CAiSE 2017, Essen, Germany, June 12-16, 2017, Proceedings*, Eric Dubois and Klaus Pohl (Eds.). Springer, 210–228. https://doi.org/10.1007/978-3-319-59536-8_14

[42] Alireza Ostovar, Abderrahmane Maaradji, Marcello La Rosa, Arthur H. M. ter Hofstede, and Boudewijn F. van Dongen. 2016. Detecting Drift from Event Streams of Unpredictable Business Processes. In *Conceptual Modeling - 35th International Conference, ER 2016, Gifu, Japan, November 14-17, 2016, Proceedings*, Isabelle Comyn-Wattiau, Katsumi Tanaka, Il-Yeol Song, Shuichiro Yamamoto, and Motoshi Saeki (Eds.). 330–346. https://doi.org/10.1007/978-3-319-46397-1_26

[43] Gyunam Park and Minseok Song. 2020. Predicting performances in business processes using deep neural networks. *Decis. Support Syst.* 129 (2020). https://doi.org/10.1016/j.dss.2019.113191

[44] Mirko Polato. 2017. Dataset belonging to the help desk log of an Italian Company. (7 2017). https://doi.org/10.4121/uuid:0c60edf1-6f83-4e75-9367-4c63b3e9d5bb

[45] Qinglin Qi and Fei Tao. 2018. Digital Twin and Big Data Towards Smart Manufacturing and Industry 4.0: 360 Degree Comparison. *IEEE Access* 6 (2018), 3585–3593. https://doi.org/10.1109/ACCESS.2018.2793265

[46] Florian Richter and Thomas Seidl. 2017. TESSERACT: Time-Drifts in Event Streams Using Series of Evolving Rolling Averages of Completion Times. In *Business Process Management - 15th International Conference, BPM 2017, Barcelona, Spain, September 10-15, 2017, Proceedings*, Josep Carmona, Gregor Engels, and Akhil Kumar (Eds.). Springer, 289–305. https://doi.org/10.1007/978-3-319-65000-5_17

[47] Williams Rizzi, Chiara Di Francescomarino, Chiara Ghidini, and Fabrizio Maria Maggi. 2022. How do I update my model? On the resilience of Predictive Process Monitoring models to change. *Knowl. Inf. Syst.* 64, 5 (2022), 1385–1416. https://doi.org/10.1007/s10115-022-01666-9

[48] Denise Maria Vecino Sato, Sheila Cristiana De Freitas, Jean Paul Barddal, and Edson Emílio Scalabrin. 2022. A Survey on Concept Drift in Process Mining. *ACM Comput. Surv.* 54, 9 (2022), 189:1–189:38. https://doi.org/10.1145/3472752

[49] Jeffrey C Schlimmer and Richard H Granger. 1986. Beyond incremental processing: tracking concept drift. In *AAAI*. 502–507.

[50] Alexander Seeliger, Timo Nolle, and Max Mühlhäuser. 2017. Detecting Concept Drift in Processes using Graph Metrics on Process Graphs. In *Proceedings of the 9th Conference on Subject-oriented Business Process Management, S-BPM ONE 2017, Darmstadt, Germany, March 30-31, 2017*, Max Mühlhäuser and Cornelia Zehbold (Eds.). ACM, 6. http://dl.acm.org/citation.cfm?id=3040566

[51] Arik Senderovich, Matthias Weidlich, Liron Yedidsion, Avigdor Gal, Avishai Mandelbaum, Sarah Kadish, and Craig A Bunnell. 2016. Conformance checking and performance improvement in scheduled processes: A queueing-network perspective. *Information Systems* 62 (2016), 185–206. https://doi.org/10.1016/j.is.2016.01.002

[52] Junming Shao, Zahra Ahmadi, and Stefan Kramer. 2014. Prototype-based learning on concept-drifting data streams. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani (Eds.). ACM, 412–421. https://doi.org/10.1145/2623330.2623609

[53] Padhraic Smyth and Rodney M. Goodman. 1991. Rule Induction Using Information Theory. In *Knowledge Discovery in Databases*, Gregory Piatetsky-Shapiro and William J. Frawley (Eds.). AAAI/MIT Press, 159–176.

[54] Xiuyao Song, Mingxi Wu, Christopher M. Jermaine, and Sanjay Ranka. 2007. Statistical change detection for multi-dimensional data. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, California, USA, August 12-15, 2007*, Pavel Berkhin, Rich Caruana, and Xindong Wu (Eds.). ACM, 667–676. https://doi.org/10.1145/1281192.1281264

[55] Ward Steeman. 2014. BPI Challenge 2013. https://doi.org/10.4121/uuid:a7ce5c55-03a7-4583-b855-98b86e1a2b07

[56] Florian Stertz and Stefanie Rinderle-Ma. 2018. Process Histories - Detecting and Representing Concept Drifts Based on Event Streams. In *On the Move to Meaningful Internet Systems. OTM 2018 Conferences - Confederated International Conferences: CoopIS, C&TC, and ODBASE 2018, Valletta, Malta, October 22-26, 2018, Proceedings, Part I*, Hervé Panetto, Christophe Debruyne, Henderik A. Proper, Claudio Agostino Ardagna, Dumitru Roman, and Robert Meersman (Eds.). Springer, 318–335. https://doi.org/10.1007/978-3-030-02610-3_18

[57] Florian Stertz and Stefanie Rinderle-Ma. 2019. Detecting and Identifying Data Drifts in Process Event Streams Based on Process Histories. In *Information Systems Engineering in Responsible Information Systems - CAiSE Forum 2019, Rome, Italy, June 3-7, 2019, Proceedings*, Cinzia Cappiello and Marcela Ruiz (Eds.). Springer, 240–252. https://doi.org/10.1007/978-3-030-21297-1_21

[58] Abhijit Suprem, Joy Arulraj, Calton Pu, and João Eduardo Ferreira. 2020. ODIN: Automated Drift Detection and Recovery in Video Analytics. *Proc. VLDB Endow.* 13, 11 (2020), 2453–2465. http://www.vldb.org/pvldb/vol13/p2453-suprem.pdf

[59] Gabriel Marques Tavares, Paolo Ceravolo, Victor G. Turrisi da Costa, Ernesto Damiani, and Sylvio Barbon Junior. 2019. Overlapping Analytic Stages in Online Process Mining. In *2019 IEEE International Conference on Services Computing, SCC 2019, Milan, Italy, July 8-13, 2019*, Elisa Bertino, Carl K. Chang, Peter Chen, Ernesto Damiani, Michael Goul, and Katsunori Oyama (Eds.). IEEE, 167–175. https://doi.org/10.1109/SCC.2019.00037

[60] Niek Tax, Ilya Verenich, Marcello La Rosa, and Marlon Dumas. 2017. Predictive Business Process Monitoring with LSTM Neural Networks. In *Advanced Information Systems Engineering - 29th International Conference, CAiSE 2017, Essen, Germany, June 12-16, 2017, Proceedings*. Springer, 477–492. https://doi.org/10.1007/978-3-319-59536-8_30

[61] Manoj Kumar M. V., Likewin Thomas, and Annappa Basava. 2015. Capturing the Sudden Concept Drift in Process Mining. In *Proceedings of the International Workshop on Algorithms & Theories for the Analysis of Event Data, ATAED 2015, Brussels, Belgium, June 22-23, 2015 (CEUR Workshop Proceedings)*, Wil M. P. van der Aalst, Robin Bergenthum, and Josep Carmona (Eds.), Vol. 1371. CEUR-WS.org, 132–143. http://ceur-ws.org/Vol-1371/paper11.pdf

[62] Wil M. P. van der Aalst. 2016. *Process mining: Data science in action*. Springer. https://doi.org/10.1007/978-3-662-49851-4

[63] Wil M. P. van der Aalst, T. Weijters, and L. Maruster. 2004. Workflow mining: discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering* 16, 9 (2004), 1128–1142. https://doi.org/10.1109/TKDE.2004.47

[64] Wil M. P. van der Aalst et al. 2011. Process Mining Manifesto. In *BPM Workshops*. Springer, 169–194. https://doi.org/10.1007/978-3-642-28108-2_19

[65] Boudewijn van Dongen. 2011. Real-life event logs - Hospital log. (3 2011). https://doi.org/10.4121/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffcf54

[66] Boudewijn van Dongen. 2015. BPI Challenge 2015. https://doi.org/10.4121/uuid:31a308ef-c844-48da-948c-305d167a0ec1

[67] Barbara Weber, Manfred Reichert, and Stefanie Rinderle-Ma. 2008. Change patterns and change support features - Enhancing flexibility in process-aware information systems. *Data Knowl. Eng.* 66, 3 (2008), 438–466. https://doi.org/10.1016/j.datak.2008.05.001

[68] A. Weijters, Wil M. P. van der Aalst, and Alves Medeiros. 2006. Process Mining with the Heuristics Miner-algorithm. *CIRP Annals - Manufacturing Technology* 166 (01 2006).

[69] Sven Weinzierl, Sandra Zilker, Matthias Stierle, Martin Matzner, and Gyunam Park. 2020. From predictive to prescriptive process monitoring: Recommending the next best actions instead of calculating the next most likely events. In *Wirtschaftsinformatik (Zentrale Tracks)*. 364–368. https://doi.org/10.30844/wi_2020_c12-weinzierl

[70] Anton Yeshchenko, Claudio Di Ciccio, Jan Mendling, and Artem Polyvyanyy. 2019. Comprehensive Process Drift Detection with Visual Analytics. In *ER*. 119–135. https://doi.org/10.1007/978-3-030-33223-5_11

[71] Anton Yeshchenko, Claudio Di Ciccio, Jan Mendling, and Artem Polyvyanyy. 2021. Visual drift detection for sequence data analysis of business processes. *IEEE Transactions on Visualization and Computer Graphics* (2021).

[72] Canbin Zheng, Lijie Wen, and Jianmin Wang. 2017. Detecting Process Concept Drifts from Event Logs. In *On the Move to Meaningful Internet Systems. OTM 2017 Conferences - Confederated International Conferences: CoopIS, C&TC, and ODBASE 2017, Rhodes, Greece, October 23-27, 2017, Proceedings, Part I*, Hervé Panetto, Christophe Debruyne, Walid Gaaloul, Mike P. Papazoglou, Adrian Paschke, Claudio Agostino Ardagna, and Robert Meersman (Eds.). Springer, 524–542. https://doi.org/10.1007/978-3-319-69462-7_33