Time-aware Concept Drift Detection Using the Earth Mover's Distance

Tobias Brockhoff[0000-0002-6593-9444], Merih Seran Uysal[0000-0003-1115-6601],
Wil M.P. van der Aalst[0000-0002-0955-6940]
Process and Data Science Group (PADS)
Computer Sience 9, RWTH Aachen University, Aachen, Germany
{brockhoff, uysal, wvdaalst}@pads.rwth-aachen.de

***Abstract*—Modern business processes are embedded in a complex environment and, thus, subjected to continuous changes. While current approaches focus on the control flow only, additional perspectives, such as time, are neglected. In this paper, we investigate a more general concept drift detection framework that is based on the Earth Mover's Distance. Our approach is flexible in terms of incorporating additional perspectives thanks to the capability of defining custom feature representations, as well as expressive feature similarity measures. We demonstrate the former by incorporating the time perspective using both a time-binning-based trace descriptor and a suitable similarity measure that considers time and control flow. We evaluate the resulting sliding window detector on different types of control-flow and time drifts, and holistic drifts involving multiple perspectives.**

***Index Terms*—Process Mining, Concept Drift Detection, Earth Mover's Distance**

I. Introduction

Due to changing conditions in business environments, business processes are continuously evolving and are, thus, seldomly in a steady state. Therefore, flexibility, supported by efficient and holistic change detection and management approaches, is an important competitive advantage and can even be necessary to persist in a competitive environment.

A major challenge for detecting concept drift is its multifactorial causes and effects. For example, new regulations change the control flow, the workload is adapted to changing market demand/price, the organization of the individual work changes over time, or new employees temporarily reduce the performance of specific parts of the process. Hence, concept drift detection methods that allow for different perspectives onto the process are needed.

To this end, this paper proposes a drift detection method based on the Earth Mover's Distance (EMD) [1] which allows us to incorporate different perspectives. Introduced in the computer vision domain as a similarity measure matching human perception well, EMD has recently gained interest in the process mining community as a method for conformance checking on stochastic languages [2]. Given two distributions and a measure of pairwise feature similarity, EMD describes the minimum effort that is required to transform one distribution into the other. Its intuitive interpretation and flexibility in terms of specifying a feature distribution and similarity measure makes EMD a general comparison method that has been applied in many different domains.

In contrast to most of the existing work which solely focuses on control-flow level drifts by means of hypothesis tests on control flow feature distributions, we propose to exploit the flexibility of EMD to address other manifestations of drift. For instance, from the resource perspective, the frequency that a certain resource executes a sequence of activities might change, while these activities are still handled by other resources. Moreover, the control-flow level drift detection does not consider *time* which is a major driver for key performance indicators. For example, an increasing case arrival rate will ultimately impact the sojourn times if the available resources become overloaded. Furthermore, the service times of certain activities, as a proxy for case complexity, can determine how the case will be further handled and, therefore, impact the general control flow. These use cases require methods that extend the general control-flow perspective and also consider time, resources or combinations of the former perspectives yielding a broader or even holistic view on potential changes.

Using EMD we demonstrate how a *time* perspective can be incorporated into concept drift detection, e.g., activity service and sojourn times. To achieve this, we use both a time-aware trace descriptor and a suitable similarity measure.

Our contributions are as follows: First, we introduce a sliding window-based concept drift detection approach that uses stochastic languages to represent the windows. Then, we use EMD to compare two windows with each other and detect a (possible) concept drift between them. Moreover, we demonstrate how the basic detection approach can be modified in order to incorporate time features.

The remainder of the paper is organized as follows: We review the related work in Section II. Then, Section IV-A introduces EMD for stochastic languages and Section IV-B demonstrates how time can be incorporated into the detection approach. After presenting the sliding windows method in Section IV-C, we evaluate our approach in Section V. Finally, we conclude our paper and give an outlook on future work in Section VI.

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.

II. Related Work

Originally, the topic of *concept drift* originated from the fields of data mining and machine learning where it describes a change of a target concept [3], [4] dependent on a hidden context. As summarized in [5], there are three major approaches for detecting concept drift. Approaches based on *sequential analysis* incrementally compute and maintain a test statistic on distribution changes of the input data, e.g., whether the probability ratio of certain subsequences changes [6]. Methods based on *statistical process control* maintain a model of the process and detect change points by monitoring prediction quality metrics [7], [8]. Finally, *window-based drift detectors* compare the empirical data distributions between two data windows at different periods in time. In this context, differences between the distributions can be assessed directly by statistical tests [9], assuming equal distributions as the null hypothesis, or by significant changes of information theoretic measures, e.g., Kullback-Leibler divergence [10].

In process mining, *window-based concept drift detectors* are the most prominent type of detector. The first approach specifically tailored to business processes was proposed in [11]. The authors apply two fixed-size sliding windows representing traces by feature vectors proposing two global and two local features, respectively. Differences between the feature vector distributions are then assessed by a hypothesis test. Finally, sliding the windows over the log yields a p-value plot that can be used for visual concept drift detection. A more detailed analysis of this approach can be found in [12]. Moreover, using improved windowing techniques, this work was automatized and extend in [13] to deal with gradual drift and multi-order dynamics, i.e., multiple changes of different granularities. A similar window-based approach representing traces by means of a trace abstraction in terms of so-called *runs* was proposed in [14]. Based on the classical α-relations [15], distributions over trace abstractions are computed and compared using a hypothesis test. The idea of using α-relations for drift detection was also extended to drift detection and characterization in event streams [16], [17]. The authors in [18] propose to compute pair-wise intra-trace activity distances over a reference window where distance interval changes indicate change points. A window-based approach that chooses a different window representation based on convex polyhedra over the Parikh vector [19] can be found in [20]. Finally, window comparison using the canberra distance on the dependency relations of the heuristic miner was proposed in [21].

In contrast to the window-based approaches, a *sequential analysis* based approach, that maintains statistics on whether directly-follows and eventually-follows relations always, never, or sometimes hold within sequential sections of the process, was introduced in [22]. Nevertheless, all these approaches only consider the control flow level.

Clustering is another approach for detecting changes. The authors in [23] propose to enhance the trace representation by a time dimension by adding the start timestamp of the trace. Although this explicitly incorporates time into the representation, time information of individual activities is not considered. A more general framework for trace clustering based change point detection is introduced in [24]. Traces within a chosen window are clustered using the Markov Cluster algorithm and the trace of the difference matrix between two cluster matrices is proposed as a window similarity measure. The downside of this approach is that control-flow features and time information are not considered by the authors. Moreover, cluster differences were only evaluated visually.

III. Prelimiaries

Before proposing our novel concept drift detection method, this section introduces the basic concepts needed in Section IV.

a) Event Log: Let Σ_A denote a finite alphabet of activities and let Σ_V be the set of (countable infinite) additional activity descriptor values that define a view on the process, e.g., $\Sigma_V = \mathbb{N}$ for temporal information based on binning (cf. Section IV-B). Each event can then be described by an activity and its additional descriptor value yielding an alphabet $\Sigma = \Sigma_A \times \Sigma_V$. Given a descriptor $e = (a, v) \in \Sigma$, $e_a = a$, $e_v = v$ denote the corresponding activity and view value, respectively. Subsequently, the set Σ^* of all finite words over Σ describes the possible *trace variants* of the process. For a trace $\sigma = (e_1, e_2, \ldots, e_n) \in \Sigma^n$, we denote the i-th element by $\sigma_i = e_i$ and the subsequence $(e_i, \ldots e_j)$ by $\sigma[i, j], i \leq j$. Using the potentially extended alphabet, the input of our method is an event log that is defined as follows:

Definition 1 (Event Log)**.** *Let Σ denote a (countable infinite) alphabet. An* event log *is a finite multiset of traces $E \colon \Sigma^* \to \mathbb{N}$.*

For example, $E = [\langle (a,1),(b,4)\rangle^7, \langle (a,4),(b,2)\rangle^3, \langle (b,2),(a,4)\rangle^1]$ is an event log containing 10 traces comprising an additional integer descriptor.

b) Stochastic Language: Stochastic languages can be used to represent collections of traces. In addition to the *trace variants* that are present, a *stochastic language* also captures the likelihood of a variant. Formally, it is defined as a function that assigns a probability to each trace.

Definition 2. *Given an alphabet Σ, $f \colon \Sigma^* \to [0, 1]$ is a stochastic language iff $\Sigma_{t \in \Sigma^*} f(t) = 1$.*

Each multiset of traces, e.g., an event log, can be transformed into a *stochastic language* by normalization. For instance, the preceding example event log can be normalized to $[\langle (a,1),(b,4)\rangle^{\frac{6}{10}}, \langle (a,4),(b,2)\rangle^{\frac{3}{10}}, \langle (b,2),(a,4)\rangle^{\frac{1}{10}}]$. The support of a *stochastic language* f will be denoted by $S_f = \{t \in \Sigma^* | f(t) > 0\}$. In the following, we will consider *finite stochastic languages*, i.e., *stochastic languages* with finite support.

IV. Methods

In this section, we propose an approach for time-aware concept drift detection. To this end, we first introduce the Earth

Mover's Distance (EMD) as a comparison method for probability distributions. Regarding the importance of incoporating additional perspectives for holistic change point detection, using the proposed method, we demonstrate how the detector can naturally be extended to consider time. Finally, we briefly describe the sliding window approach that was used to make the method applicable for concept drift detection.

A. Concept Drift Detection based on EMD

The Earth Mover's Distance (EMD) is a distance-based similarity measure for the comparison of probability distributions. Intuitively, it considers one distribution as piles of earth and the other as a set of holes. Given a ground distance function that describes the effort of moving earth from a certain pile to a specific hole, EMD is the minimum-cost flow required to move earth from hills to holes.

a) Trace Distance: Before formally defining EMD as a distance measure for stochastic languages, we introduce the notion of a trace distance [2]. A trace distance $\delta\colon \Sigma^* \times \Sigma^* \to [0,1]$ describes the dissimilarity of any two given traces. It is required to be symmetric and to satisfy the identity of indiscernibles porperty [2]. Besides, it is desirable that δ additionally satisfies the triangle inequality, i.e., $\forall t, t', t'' \in \Sigma^*\colon \delta(t,t') + \delta(t',t'') \leq \delta(t,t'')$, so that δ, and therefore also EMD, become a metric. Nevertheless, as detailed in Section IV-B, some trace distance candidates might only empirically satisfy the triangle inequality for most inputs.

b) EMD for Stochastic Languages: Given a trace distance function, EMD as a measure for comparing stochastic languages has been proposed in [2]. In general, EMD as a similarity measure between probability distributions unites two desirable properties. On the one hand, it is frequency-aware in the sense that it considers the magnitude of discovered differences. On the other hand, the notion of difference is determined by the trace distance, and, thus, different distance function can express different perceptions of similarity. In the context of processes, we use this property to open different perspectives on the process. Formally, in the context of finite stochastic languages f, g, EMD between f and g can be defined by the following linear program:

Definition 3 (Earth Mover's Distance)**.** *Let f, g be finite stochastic languages over the alphabet Σ and let δ be a trace distance function. The Earth Mover's Distance between f and g is defined by:*

$$EMD(f,g) = \textit{minimize} \quad \sum_{t \in S_f} \sum_{u \in S_g} x_{tu} \cdot \delta(t,u)$$

$$\textit{s.t.} \qquad \forall t \in S_f : \sum_{u \in S_g} x_{tu} = f(t) \quad \textit{(Source)}$$

$$\forall u \in S_g : \sum_{t \in S_f} x_{tu} = g(u) \quad \textit{(Target)}$$

$$\forall t \in S_f \, \forall u \in S_g : \quad x_{tu} \geq 0 \quad \textit{(Non-negativity)}.$$

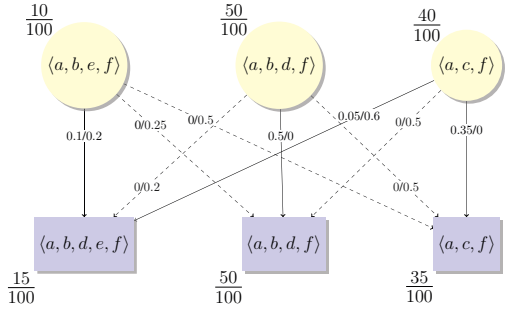


Fig. 1: Optimal cost flow for $EMD(f_1, f_2)$ for the stochastic languages f_1, f_2. Edge inscriptions $x^*_{tu}/\delta(t,u)$ show the optimal flow x^*_{tu} and the trace distance value $\delta(t,u)$ for all variants $t \in S_{f_1}, u \in S_{f_2}$. Zero flow edges are depicted by dashed lines. Even though we have $S_{f_1} \neq S_{f_2}$, $EMD(f_1, f_2) = 0.05$ shows that the difference concerns infrequent behavior.

While the last constraints ensure non-negative flows, the source constraints assert that the entire probability mass for each trace of f is moved to traces in g. Likewise, the target constraints limit the flows to g according to its probability mass distribution. Furthermore, the objective function ensures a minimal effort flow. Note that, as we consider finite stochastic languages, the number of variables and constraints is finite, and, thus, the linear program is well-defined. Moreover, the linear program is feasible due to the equality of the total weight of f and g as stochastic languages are inherently normalized. Besides, normalization and a metric as trace distance would make EMD on stochastic languages a metric too [1].

To detect control flow changes between two stochastic languages, we can for example use the post-normalized Levenshtein distance, where in a second step the computed distance value is normalized by the maximum length of the two considered sequences, as trace distance function. For instance, consider the stochastic languages $f_1 = [\langle a,b,d,f\rangle \frac{50}{100}, \langle a,c,f\rangle \frac{40}{100}, \langle a,b,e,f\rangle \frac{10}{100}]$, $f_2 = [\langle a,b,d,f\rangle \frac{50}{100}, \langle a,c,f\rangle \frac{35}{100}, \langle a,b,d,e,f\rangle \frac{15}{100}]$ and $f_3 = [\langle a,b,d,f\rangle \frac{20}{100}, \langle a,c,f\rangle \frac{70}{100}, \langle a,b,e,f\rangle \frac{10}{100}]$ where f_1 represents the basic process behavior. While in f_2 an infrequent new variant where both d and e can occur is introduced replacing another infrequent variant, in f_3 the most frequent variant changes from $\langle a,b,d,f\rangle$ to $\langle a,c,f\rangle$. Using the (post)normalized Levenshtein distance, we obtain $EMD(f_1, f_2) = 0.05$ and $EMD(f_1, f_3) = 0.15$ which well reflects the major behavior change in f_3. The example solution to the linear program for $EMD(f_1, f_2)$ is depicted in Figure 1. It shows that, on the one hand, the costs are caused by transferring a probability mass of 0.1 between the two infrequent variants $\langle a,b,e,f\rangle$ and $\langle a,b,d,e,f\rangle$, which have a post-normalized trace distance of 0.2. On the other hand, a slightly reduced likelihood of $\langle a,c,f\rangle$ in f_2 causes additional costs of $0.05 \cdot 0.6 = 0.03$. Using EMD for stochastic language comparison enables us to focus on trace properties by means of the trace distance function. Moreover, it also allows us to consider the frequency of observed patterns.

B. Time-Aware Concept Drift Detection

In this section, we propose to extend concept drift detection by adding a time perspective, i.e., sojourn times or activity service times. To this end, we exploit the flexibility of EMD and introduce a time-aware ground distance. We first bin the observed times and then apply a weighted Levenshtein distance variant in order to obtain a control flow and time-aware comparison method.

a) Binning: We apply time binning in order to abstract similar time related behavior and to make the weighted Levenshtein distance applicable. To this end, depending on the chosen time perspective, first, we collect the observed activity service or sojourn times. As time features might differ significantly between different activities, we locally evaluate the following two clustering approaches, i.e., we compute the statistics per activity while taking the complete log into account. Following the Pareto Principle that 80% of the interesting behavior is caused by 20% of the cases, we propose to determine the bins by a percentile-based clustering based on the times observed for each individual activity. The bin edges are thereby chosen according to a given list of percentiles. Regarding multimodal time distributions we propose to use the k-means++ [25] as a data-driven clustering approach. In the 1-dimensional time setting its objective is theoretically well-founded being equivalent to minimizing the within-class variance [26]. A major advantage is that this approach can deal with different cluster sizes among different activities. By applying these binning techniques, we discretize time-related information into k bins and obtain traces over an alphabet $\Sigma_T = \Sigma_A \times \{0, \dots, k-1\}$ with the bin index as an additional descriptor.

b) Time-Aware Trace Distance: Given two traces $\sigma, \sigma' \in \Sigma_T^*$ over the binned alphabet with k bins, we propose to use a variant of the weighted Levenshtein distance δ_l [27] which splits the total operation costs into control-flow and time costs, respectively. To this end, on the control-flow level, we define binary costs $c_r^f \colon \Sigma_A \times \Sigma_A \to \mathbb{R}^{\geq 0}$ for renaming and unary costs $c_{\text{id}}^f \colon \Sigma_A \to \mathbb{R}^{\geq 0}$ for insertion and deletion. Likewise, on the time level, we use binary costs $c_{\text{mr}}^t \colon \{0, \dots, k-1\} \times \{0, \dots, k-1\} \to \mathbb{R}^{\geq 0}$, for matching and renaming activities considering different time bins and unary costs $c_{\text{id}}^t \colon \{0, \dots, k-1\} \to \mathbb{R}^{\geq 0}$ for insertion and deletion. The resulting trace distance can then be defined by the following recursive relation:

$$\delta_l(\sigma[1,i],\sigma'[1,j])= \min \begin{cases} \delta_l(\sigma[1,i-1],\sigma'[1,j-1]) + c_{\text{mr}}^t((\sigma_i)_v,(\sigma'_j)_v) & \text{if}(\sigma_i)_a=(\sigma'_j)_a \\ \delta_l(\sigma[1,i-1],\sigma'[1,j-1]) + & \\ \quad c_r^f((\sigma_i)_a,(\sigma'_j)_a) + c_{\text{mr}}^t((\sigma_i)_v,(\sigma'_j)_v) & \text{(Rename)} \\ \delta_l(\sigma[1,i],\sigma'[1,j-1]) + c_{\text{id}}^f((\sigma'_j)_a) + c_{\text{id}}^t((\sigma'_j)_v) & \text{(Insert)} \\ \delta_l(\sigma[1,i-1],\sigma'[1,j]) + c_{\text{id}}^f((\sigma_i)_a) + c_{\text{id}}^t((\sigma_i)_v) & \text{(Delete)}\,. \end{cases} \tag{1}$$

Regarding a concrete instantiation of the cost functions, we need to ensure that it respects the inter-dependencies between the edit operations to preserve the intuitive interpretability of the distance value which is stated in the following axiom.

Axiom 1 (Levenshtein-based Trace Distance Interpretability)**.** *Levenshtein-based trace distances that consider time information should fulfill the following properties:*

1. *Renaming is preferable over deletion and insertion*
2. *Matching events under a large time difference is less expensive than*
   (a) *deletion followed by insertion*
   (b) *renaming events with different activity labels but the same time information.*

Moreover, since we will use δ_l in a sliding-window framework, we want δ_l to be symmetric so that traces from the left and right window are treated equally. For example, consider three subsequent windows each containing just a single trace variant, namely σ^1, σ^2 and σ^3, such that the first and third window are equal, i.e., $\sigma^1 = \sigma^3$. Intuitively, differences and, hence, potential drift points between the first and the second, or between the second and the third window should be scored equally, i.e., $\delta_l(\sigma, \sigma') = \delta_l(\sigma', \sigma'')$. Therefore, we use the same function for the insertion and deletion cost to make δ_l symmetric.

Considering the interpretation axioms, let $a, a' \in \Sigma_A, a \neq a'$, $t, t' \in \{0, \dots, k-1\}$. We use fixed control-flow penalties for renaming, insertion, and deletion, i.e., $c_r^f(a, a') = f_r = 1 = f_{\text{id}} = c_{\text{id}}^f(a)$, costs linear in the bin distance for matching and renaming w.r.t. time, i.e., $c_m^t(t, t') = c_r^t(t, t') = |t - t'|$, and cost linear in the bin index for insertion and deletion, i.e., $c_{\text{id}}^t(t) = t$. Even though time bins of disjoint activities are not necessarily comparable w.r.t. absolute time, this choice naturally enforces Axiom 1.2(b). For example, consider the costs and traces $f_r = 2, f_{\text{id}} = 2$, $\sigma = \langle (a,1),(b,4) \rangle$, $\sigma' = \langle (a,4),(b,2) \rangle$ and $\sigma'' = \langle (b,2),(a,4) \rangle$. If renaming can only be applied to different activity labels, is not per se expensive and does not respect time differences, we obtain $3 + 2 = \delta_l(\sigma, \sigma') \geq \delta_l(\sigma, \sigma'') = 2 + 2$ although renaming is required twice in the latter case. This contradicts the intuitive perception of similarity. Likewise, for high renaming costs, that are independent of the time difference, renaming operations would dominate EMD, therefore, rendering it less sensitive to time behavior changes. In addition to a general operation penalty, we also consider the time bin for insertion and deletion operations enforcing Axiom 1.2(a).

In order to account for different trace lengths, we post-normalize the distance by the maximum trace lengths, i.e., $\bar{\delta}_l(\sigma, \sigma') = \frac{\delta_l(\sigma,\sigma')}{\max(|\sigma|,|\sigma'|)}, \sigma, \sigma' \in \Sigma^*$. Even though δ_l is a metric, normalization might violate the triangle inequality for Levenshtein based edit distances [28]. Accordingly, $\bar{\delta}_l$ is not necessarily a metric. Nevertheless, the triangle inequality and thus metric properties emprically hold for most of the descriptors considered in our evaluation.

C. Sliding Window

In order to detect concept drift in an event log, we embed our approach into a sliding window framework. Let $(t_1, t_2, \dots, t_n) \in (\Sigma^*)^n$ denote an ordering of the traces

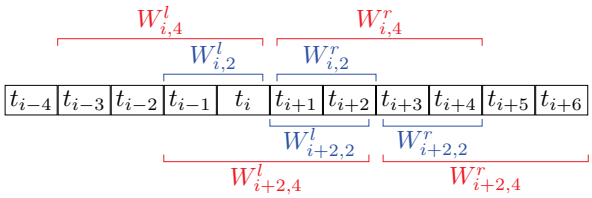


Fig. 2: Multi-scale sliding window. At each scale, viz. window size 2 and 4, we maintain a left and right window. After comparison, both windows are shifted by a stride size of 2.

by the timestamp of the first event. As depicted in Figure 2, given an index i, $W^l_{i,w} = (t_{i-w+1}, t_{i-w}, \ldots, t_i)$ and $W^r_{i,w} = (t_{i+1}, t_{i+2}, \ldots, t_w)$ denote the left and right window, respectively. We compute the corresponding trace descriptors for each window pair $W^l_{i,w}$ and $W^r_{i,w}$ and transform the resulting representations into stochastic languages f^l_i and f^l_r by normalization. Afterwards, we compare f^l_i and f^l_r using EMD yielding the dissimilarity score $EMD(f^l_i, f^r_i)$. Finally, the windows are shifted by a user-defined stride size. Increasing the stride size, thereby trades localization accuracy for efficiency. We plot the final EMD values to detect process changes. Similar to the work in [29], we use different-sized windows in order to show and validate drift on multiple scales.

V. Evaluation

In this section, we demonstrate the capability of the proposed method to detect control-flow drifts, time drifts, and control-flow drifts which can only be detected by a holistic approach. To this end, we created multiple artificial event logs from variants of the model used in [11]. Figure 3 depicts the adapted model, which models the process of handling health insurance claims, in BPMN notation. Arriving claims are registered and classified into low and high claims. Besides, a questionnaire is sent to the claimant which might be answered. Each claim undergoes several checks which either lead to rejection or eventually to acceptance. In contrast to the original model, our model adapts the efficient *reject-if-check-is-unsuccessful* pattern for low and high claims. Furthermore, our model introduces refunding activities before the claimant is notified. After checking, every claimant receives a documented notification and can be called additionally. Finally, the case is archived while some cases are also controlled in more detail. In order to assign the tasks to resources, we maintain a resource pool and randomly choose free resource therefrom. Following natural behavior, we sample exponentially distributed inter-case arrival times while the service times of human executed activities are normal distributed.

A. Implementation

We implemented the methods presented in Section IV as a plugin in ProM. The plugin allows to specify multiple sliding windows of different window and stride sizes. Moreover, it supports the Levenshtein distance and the weighted Levenshtein distance for time-binning based trace descriptors, that we proposed in Section IV-B, as trace distance functions. Regarding the time perspective, the user can choose between service and sojourn time-aware concept drift detection. In this case, the time bins are computed according to user-defined percentiles for the percentile-based bin edge computation or a given number of clusters for the data-driven approach using k-means. In order to efficiently compute EMD, we implemented an Exterior Point Simplex method described in [30], [31], which is tailored to the underlying optimal-cost-flow problem.

B. Control-Flow Drift

In order to demonstrate the ability of our approach to detect control-flow drifts, we apply it to the log used in [11]. The log contains different types of control-flow drift with a change every 1200 traces. Figure 4a shows the output of our plugin for different window sizes using EMD with Levenshtein distance for window comparison. The bottom panel shows a heatmap of EMD values w.r.t. the indices of the traces, ordered by the timestamp of the first event, and the window sizes. Given that different window sizes capture changes of different frequencies, this spectrum-inspired visualization can help to localize changes in the time and frequency domain. The upper panel shows a lineplot of the EMD values for an interactively selectable window size.

First, as depicted by the dotted lines, we note that all model changes can be detected, although the inherently high-dimensional trace descriptor requires large windows sizes to capture the process behavior. As can be seen for window size 200, the significantly higher variance indicates that the sample size is insufficient to fully represent the underlying distribution. In this case, larger windows are needed to verify potential drift points. Moreover, it can be seen that EMD naturally describes the extend to which the process is affected by the change. To this end, consider the highlighted EMD value difference between the second and third peak. While the pre-change model for the second peak contains an or-relation between 'By Phone', 'E-Mail', and 'Post', as well as the option to skip everything, the change removes the latter option. The low EMD values thereby indicate that relatively few cases are affected, i.e., the notification has been skipped for these cases. In contrast, the third peak results from introducing an exclusive choice between the former activities. Considerably larger EMD values show that many cases are affected, i.e., before the change claimants were often notified via multiple channels.

Although the currently implemented window descriptor based on stochastic languages, which consider entire traces, can be effectively high-dimensional, an advantage is that we can detect long-term dependencies. In order to show this, we introduce a controlling activity scrutinizing a limited subset of the recently archived cases. Initially, given two low and high claim cases, the probability that the high claim case is selected for additional controlling is 40%. After 2000 cases this probability changes to 20%, following the idea that high claim cases have been handled more carefully, and therefore less controlling is required. This introduces a long-term dependency between the initial check claim sequence

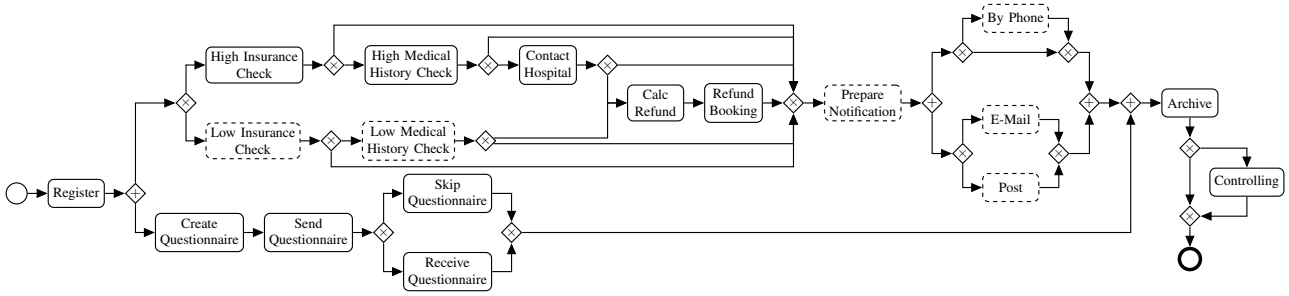
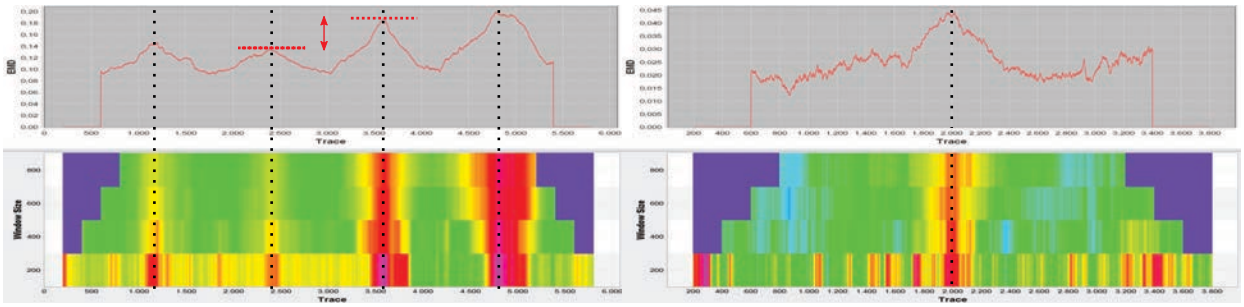


Fig. 3: The adapted insurance claim model from [11] showing a process for handling low and high insurance claims. Dashed lines show the activities that are affected by the filtering applied in the time-induced control flow drift scenario.



(a) Different control-flow drifts (stride size 2).

(b) Long-term dependency drift (stride size 2).

Fig. 4: Drift detection on the control-flow level using the Levenshtein distance. It shows the spectrum over window sizes 200, 400, 600, 800 and selectively EMD for window size 600.

and the controlling activity which is further extended by the introduction of the refunding activities (cf. Figure 3). As depicted in Figure 4b, the drift can be detected which would not be possible using local statistics, e.g., a window around each activity if the window size is too small [11]. Intuitively, such drift scenarios are relevant for activities whose number of occurrences is limited by the resource availability and, therefore, drifts do not necessarily affect the follows frequencies of directly preceding activities.

C. Time Drift

In order to assess the capability to detect time drift, we inject changes of the case arrival rate. Using an initial base inter-arrival time of 45 minutes, we created a log with sudden arrival rate drifts to 60, 45, 30, 45, 25, and 45 minutes every 2000 traces, respectively. In order to exclude the possibility of inter-arrival-time-induced control-flow drifts, we filter the *questionnaire-related activities*. Due to the increased sojourn times, more questionnaires are received back which causes a control-flow drift.

Regarding the parameter setting of f_{id} and f_r, preceding experiments showed that $f_{\text{id}} = f_r = 1$ works well. In general, we could see that the output of our method is not very sensitive to this setting for reasonably small values. Furthermore, we apply the data-driven bin computation using k-means, which generally performed well, with $k = 3$. This follows the intuition to have a bin for slow, moderate, and fast behavior.

As depicted by the dotted blue lines in Figure 5a, according to our method an increasing inter-arrival time does not significantly affect the sojourn times. During periods of baseline arrival rate, the resources are sufficient to handle each case immediately. Therefore, reducing the arrival rate does not affect the immediate reaction, and hence the sojourn times do not change. In contrast, decreasing the inter-arrival times increases the sojourn times and a drift is detected. The peaks corresponding to the onsets of arrival rate drifts are highlighted by dotted black lines. In addition to the ground truth arrival rate changes, we observe two additional peaks, depicted by the densely dotted lines, which might be attributed to the variance of the arrival rate and activity service times. Moreover, larger EMD values for shorter inter-arrival times show that EMD is a measure for the impact of the drift. Regarding our definition of the time-aware trace distance function δ_l, this experiment shows that incorporating the time bins into the costs for insertion and removal causes a shift of the baseline costs, as indicated by the dashed red line, so that the EMD values are generally larger during periods of faster case arrival. Given two traces for which the optimal distance according to δ_l requires deletion and insertion, a low arrival rate and, therefore, low sojourn times and bin indices yield a comparatively smaller distance value than a high arrival rate and consequently higher bin indices due to higher costs for insertion and deletion. Hence, the costs arising from imperfectly matching windows are generally higher. Since the baseline shifts accumulate,

we can, to some extend, also observe ongoing gradual drifts towards increasing sojourn times during the periods of arrival rate change.

D. Time-induced Control Flow Drift

Finally, we investigate a drift scenario that requires a holistic approach combining the time and control perspective. To this end, we distinguish between low claims with an easy medical history and those with a difficult history. In the model, this difference reflects in the service time of 'Low Medical History Check' with an increased service time for more difficult cases. The service times for each group are modeled by normal distributions where the probability of a low claim being difficult is 20%. The service times sampled from the resulting Gaussian mixture model are depicted in Figure 6. Assuming that the resources for calling claimants are limited and therefore only a fixed percentage of the claimants can be called, we introduce a drift by changing the individual probabilities for notification by phone for easy and difficult cases. While initially the probability is 50% for both degrees of difficulty, after the drift, that is introduced after 2500 cases, difficult cases are certainly called and easy cases only with a probability of approximately 40%. Note that in order to detect this drift, time has to be considered since the general control-flow on the activity level is not affected.

Detecting the aforementioned concept drift requires us to focus on the low claim check activities and the subsequent notification as there is only a small impact on the complete process. Therefore, we filter the generated log, containing 5000 traces, for low claim cases by only considering claims that experience the 'Low Insurance Check'. Note that cases that are rejected in this step are still considered, which slightly broadens the focus. Furthermore, we only focus on checking and notification related activities depicted by dotted lines in Figure 3. As before, we apply data-driven bin computation with $k = 3$. Figure 5b shows the output for the filtered log. The dotted line indicates a drift that occurs after half of the traces, which fits the time of the actual drift. The magnitude of EMD change also gives evidence that this drift only slightly affects the process, i.e., only a few claims are concerned. Moreover, the high variance and large EMD values that can be seen in the multi-scale plot in Figure 5b indicate a high variability of the binning-based descriptor. Therefore large sample sizes, i.e., large windows, are required to approximate the underlying distribution to obtain clearer peaks, as indicated by the black circle. Besides, it can be noted that it suffices to categorize service times into three bins to detect the drift, even though the underlying service time distribution is bi-modal.

Consequently, the presented experimental results show that our method is able to detect drifts that require a holistic consideration of time and control flow.

VI. Conclusion and Future Work

In this paper, we proposed an EMD-based concept drift detection approach that allows for different perspectives on the process thanks to the flexibility of EMD regarding the choice of the representation and the distance measure. Furthermore, we demonstrated how a time perspective that enables service and sojourn-time aware drift detection can be implemented in this framework. The experimental evaluation showed that various types of control-flow, as well as time-dependent control-flow drifts can successfully be detected. However, the ability of the currently used trace descriptor to holistically detect drift in long-term dependencies is traded for the requirement of large windows sizes and possible additional preprocessing.

For future work, given our first results on holistic drift detection, we intend to perform a more comprehensive evaluation, in particular on real-world data, as well as to develop methods that allow to systematically find a proper preprocessing without requiring domain knowledge. Considering the trace descriptors, e.g., lower-dimensional descriptors and descriptors that incorporate additional perspectives are possible future directions to investigate. Moreover, descriptors and trace distances that are more robust to concurrency can be investigated. Furthermore, since binning the time information can result in information loss, trace distance functions that do not rely on binning need to be investigated. However, first and foremost, extensions that provide drift diagnostics, i.e., methods that reveal the actual changes of the process, are crucial to make the detected drifts actionable. For example, consider the additional peaks in Figure 5a. Currently, we can only diagnose that either control flow and sojourn times are affected or by considering only the control flow perspective conclude that only the control flow is affected. Nevertheless, in neither case we can clearly pinpoint the effect. In this context, we want to exploit the optimal values of the flow variables in the EMD-defining linear program. By construction, these values determine the EMD value and, therefore, describe the differences between the stochastic languages. Finally, the application to event streams can be investigated, which, in addition to dealing with potentially changing time information granularity, requires to adapt the trace descriptor and/or distance, e.g. by allowing partial matches, to handle incomplete traces.

References

[1] Y. Rubner, C. Tomasi, and L. J. Guibas, "A metric for distributions with applications to image databases," in *ICCV*, 1998, p. 59.

[2] S. J. J. Leemans, A. F. Syring, and W. M. P. van der Aalst, "Earth movers' stochastic conformance checking," in *BPM Forum*, 2019, pp. 127–143.

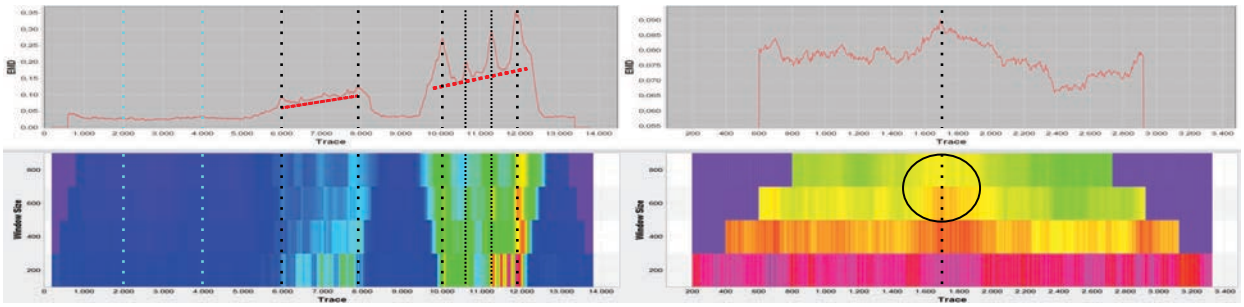
[3] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Mach. Learn.*, vol. 23, Nov. 1994.

[4] A. Tsymbal, "The problem of concept drift: Definitions and related work," Tech. Rep., 2004.

[5] J. Gama, I. Žliobaitundefined, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, Mar. 2014.

[6] A. Wald, "Sequential tests of statistical hypotheses," *Ann. Math. Statist.*, vol. 16, no. 2, pp. 117–186, Jun. 1945.

[7] A. Bouchachia, "Fuzzy classification in dynamic environments," *Soft Comp.*, vol. 15, no. 5, pp. 1009–1022, May 2011.



(a) Sojourn time drifts.

(b) Activity duration induced control-flow drift.

Fig. 5: Drift detection incorporating the time perspective. Window sizes 200, 400, 600, 800, selectively displaying EMD for window size 600. The stride size is 5.

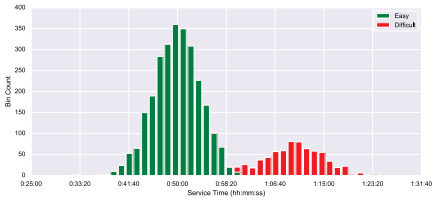


Fig. 6: Service time distribution for low claims according to the degree of difficulty.

[8] R. Klinkenberg, "Adaptive information filtering: Learning in the presence of concept drifts," in *AAAI*, 1998, pp. 33–40.

[9] D. Kifer, S. Ben-David, and J. Gehrke, "Detecting change in data streams," in *VLDB*, 2004, pp. 180–191.

[10] R. Sebastião and J. Gama, "Change detection in learning histograms from data streams," in *EPIA*, 2007, pp. 112–123.

[11] R. P. J. C. Bose, W. M. P. van der Aalst, I. Žliobaitė, and M. Pechenizkiy, "Handling concept drift in process mining," in *CAiSE*, 2011, pp. 391–405.

[12] R. P. J. C. Bose, W. M. P. van der Aalst, I. Žliobaitė, and M. Pechenizkiy, "Dealing with concept drifts in process mining," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 1, pp. 154–171, Jan. 2014.

[13] J. Martjushev, R. P. J. C. Bose, and W. M. P. van der Aalst, "Change point detection and dealing with gradual and multi-order dynamics in process mining," in *BIR*, 2015, pp. 161–178.

[14] A. Maaradji, M. Dumas, M. La Rosa, and A. Ostovar, "Fast and accurate business process drift detection," in *BPM*, 2015, pp. 406–422.

[15] W. van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: Discovering process models from event logs," *TKDE*, vol. 16, no. 9, pp. 1128–1142, 2004.

[16] A. Ostovar, A. Maaradji, M. La Rosa, A. H. M. ter Hofstede, and B. F. V. van Dongen, "Detecting drift from event streams of unpredictable business processes," in *Conceptual Modeling*, 2016, pp. 330–346.

[17] A. Ostovar, A. Maaradji, M. La Rosa, and A. H. M. ter Hofstede, "Characterizing drift from event streams of business processes," in *CAiSE*, 2017, pp. 210–228.

[18] R. Accorsi and T. Stocker, "Discovering workflow changes with time-based trace clustering," in *SIMPDA*, 2012, pp. 154–168.

[19] J. Carmona and J. Cortadella, "Process mining meets abstract interpretation," in *ECML PKDD*, 2010, pp. 184–199.

[20] J. Carmona and R. Gavaldà, "Online techniques for dealing with concept drift in process mining," in *IDA*, 2012, pp. 90–102.

[21] T. Li, T. He, Z. Wang, Y. Zhang, and D. Chu, "Unraveling process evolution by handling concept drifts in process mining," in *SCC*, 2017, pp. 442–449.

[22] C. Zheng, L. Wen, and J. Wang, "Detecting process concept drifts from event logs," in *OTM*, 2017, pp. 524–542.

[23] D. Luengo and M. Sepúlveda, "Applying clustering in process mining to find different versions of a business process that changes over time," in *BPM Workshops*, 2012, pp. 153–158.

[24] B. Hompes, J. Buijs, W. van der Aalst, P. Dixit, and J. Buurman, "Detecting change in processes using comparative trace clustering," in *SIMPDA*, 2015, pp. 95–108.

[25] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *SODA*, 2007, pp. 1027–1035.

[26] D. Liu and J. Yu, "Otsu method and k-means," in *HIS*, vol. 1, 2009, pp. 344–349.

[27] R. A. Wagner and M. J. Fischer, "The string-to-string correction problem," *JACM*, vol. 21, no. 1, pp. 168–173, Jan. 1974.

[28] A. Marzal and E. Vidal, "Computation of normalized edit distance and applications," *TPAMI*, vol. 15, no. 9, pp. 926–932, 1993.

[29] T. Dasu, S. Krishnan, S. Venkatasubramanian, and K. Yi, "An information-theoretic approach to detecting changes in multi-dimensional data streams," in *Symp. Int. Comp. Scie. Stat.*, 2006.

[30] K. Paparrizos, "A non improving simplex algorithm for transportation problems," *RAIRO-OPER RES*, vol. 30, no. 1, pp. 1–15, 1996.

[31] H. Achatz, P. Kleinschmidt, and K. Paparrizos, "A dual forest algorithm for the assignment problem," in *Applied Geometry and Discrete Mathematics: The Victor Klee Festschrift*, vol. 4, 1991, pp. 1–10.