Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 197 (2022) 751–758

Procedia Computer Science

www.elsevier.com/locate/procedia

Sixth Information Systems International Conference (ISICO 2021)

Real-time probing of control-flow and data-flow in event logs

Paolo Ceravolo[a], Ernesto Damiani[b], Emilio Francesco Schepis[a], Gabriel Marques Tavares[a,*]

[a] *Università degli Studi di Milano (UNIMI), Milan, Italy*
[b] *Khalifa University (KUST), Abu Dhabi, UAE*

Abstract

Traditional Process Mining offers batch analysis of business processes but does not transpose smoothly into online environments due to specific design constraints. Techniques adapted to support online analysis require peculiar adjustments that inherently restrict their focus to a single task. In this work, we extend the Concept Drift in Event Stream Framework (CDESF) tool to handle multiple attributes simultaneously. Our extension promotes CDESF to analyze both control-flow and data-flow characteristics in online event streams. Experiments used real and synthetic data for concept drift and anomaly detections. Results show that additional perspectives should be considered as they contain valuable information about processes.

Keywords: Online process mining; concept drift detection; event stream; clustering; anomaly detection.

1. Introduction

Information systems (IS) control, support, and regulate the execution of business processes within organizations. Process-Aware Information Systems (PAIS) are software systems that support business processes instead of only managing isolated activities [1]. Data produced by PAIS is recorded in the form of event logs, i.e. the activities executed and their associated attributes. Process Mining (PM) is the area that investigates business processes from a data standpoint, leveraging the knowledge for stakeholders [2]. Traditional PM analysis relies on *offline* processing to achieve an understanding of business processes. Offline techniques leverage event logs of completed processes as they require a multi-pass analysis of event log data. Concomitantly, organizations are seeking a real-time response for their business process executions. Fast response to environmental conditions may increase process performance and avoid resource loss. However, a challenge arises as offline solutions cannot be directly transposed to online environments due to specific constraints that regulate the process behavior, such as the need for bounding the memory size, and regularly run model updates and detection of changes [3].

In Data Stream Processing, it is assumed that streams are a continuous flow of events with an unbounded size [4]. Hence, an imminent restriction imposed is the limited amount of time and memory available. Forgetting mechanisms are required to control memory consumption and fast algorithms are needed as events must be processed faster than

* Corresponding author.
E-mail address: gabriel.tavares@unimi.it

1877-0509
10.1016/j.procs.2021.12.197

the stream arrival rate. There are several approaches to treat event streams, such as event accumulation, sliding windows and adaptive buckets [3]. Furthermore, online scenarios are susceptible to concept drifts, a phenomenon that characterizes the change of the relation between a feature and its label over time. From the PM perspective, drifts affect the validity of the process model, further harming the quality of consequent PM tasks such as conformance checking [3]. Approaches that handle specific tasks in online PM were proposed in the past mostly for online process discovery [5] [6], conformance checking [7] [8] and concept drift detection [9] [10] [11] [12] [13]. Other than limiting to a single task in online PM, current solutions usually only consider the control-flow aspect of business processes, ignoring the data-flow perspective. Control-flow refers to the sequence of executed activities within process instances. Complementarily, the data flow regards additional attributes attached to activity executions, such as the actor, associated costs, organizational information, among others.

In the context of drift detection in PM, Concept Drift in Event Stream Framework (CDESF) was proposed in [14] [15] [16]. CDESF aims at detecting drifts in business processes while also supporting several PM tasks such as online process discovery, model update, and anomaly detection. The framework ingests event streams and, using a graph structure, computes distances between current and expected behavior. Relying on an online clustering algorithm, cases are placed in the feature space and updated according to new events in the stream. The drift detection mechanism is controlled by the maintained online feature space. CDESF identifies a drift in the business process given the emergence of new behavior in the feature space. However, CDESF was limited to the control-flow and time perspectives, meaning that it only processed the activity sequence and time spans between them.

In this work, we extend CDESF to support multiple dimensions in online processing, including all data-flow attributes belonging to an event log. For that, we dynamically instantiate attribute representations in online processing. Therefore, CDESF's new version allows drift detection in the data-flow perspective. Moreover, anomalies affecting data-flow attributes are now identifiable by the upgraded CDESF. We submitted the tool to a set of experiments using real and synthetic data. The experiments aim at evaluating complementary aspects of online PM analysis, showcasing CDESF's capabilities for drift detection and overall support for real-time PM tasks. Results show that data-flow attributes are vulnerable to drifts and anomalies, indicating the need for tools that handle multiple perspectives in online PM.

The remainder of this paper is organized as follows. Section 2 introduces a detailed discussion about concept drift detection in PM, highlighting the difference between offline and online tools. Section 3 presents the CDESF workflow with special attention to the upgrades proposed in this paper. Section 4 exhibits the experiments, the employed data, and the performance metrics for evaluation. Section 5 presents the results of the experiments and discusses how additional dimensions influence CDESF's performance. Finally, Section 6 concludes the paper.

2. Related work

The recent uptake in drift detection methods proposed in PM research is motivated by the need of reacting and adapting to processes in real-time [3]. Traditionally, PM techniques rely on *offline* data processing, i.e., batch processing where multiple-pass analyses are allowed. These techniques extract information from historical executions of the business process, which may not represent the current process behavior. On the other hand, *online* PM refers to techniques prepared to ingest a stream, dynamically responding to incoming data. An additional challenge posed by streaming environments is the common non-stationarity of data. Streaming literature defines a data stream as an ordered pair (s, Δ) where s is a sequence of tuples and Δ is a sequence of positive real time-intervals [17]. Furthermore, when the true relation between a feature vector and its associated class, i.e. a *concept*, changes in two separate points in time, we identify a concept drift [4]. Properly detecting and adapting to changing concepts leverages the quality of online PM approaches. In this review, we focus on concept drift detection methods for online process data. For an in-depth analysis of online PM, we refer the reader to [3].

An early approach for online drift detection in PM was proposed in [9]. The authors extract *follow* and *precede* relations from the log and, using a windowed approach, a statistical test is performed in two non-overlapping windows. When distributions between the two windows are different, a drift alert is raised. A clustering-based approach was proposed by Zheng et al. [10]. For that, direct succession and weak order relations are extracted from traces. Candidate drift points are leveraged from the analysis of the variation trends. Then, drifts are detected by the clusterization of candidate points. Focusing on comprehensiveness, Yeshchenko et al. [11] proposed a method for drift detection and visualization. The approach divides the log into sub-logs and extracts declarative process behavior metrics based on temporal rules. Using a multivariate time series analysis, constraints confidences are extracted and clustered. Finally, each cluster is evaluated for change detection, producing a notion of behavior-specific drifts and overall drifts. Brockhoff et al. [12] proposed a method that handles both control-flow and time perspectives for drift detection. The technique is based on earth mover's distance, a distance-based similarity measure that compares probability distributions. The combination of a sliding window approach with an earth mover's distance provides a flexible approach that facilitates the detection of drifts in more than one perspective. The main limitation of previously mentioned approaches is that they perform in an offline setting, i.e., detecting drifts from historical data. Treating the event log as batch analysis (transforming the log into trace streams or sub-logs) is a constraint that hinders the application of these techniques in real scenarios, which are permeated by event streams [3]. Ostovar et al. [13] proposed a method

that ingests streams of events. The approach relies on the concept of $\alpha+$ relations, which captures several types of associations between activities [18]. Statistical tests performed in two sliding windows identify the drifts. Particular attention is given to the trade-off between accuracy and detection latency, which is a constraint in online processing [3]. However, the approach ignore other data dimensions that could also be affected by concept drifts.

3. Multi-dimensions in Online Process Mining

CDESF is a multi-purpose tool to support the analysis of event streams. Although the main goal is drift detection, CDESF also supports online process discovery and anomaly detection. The framework was first presented in [14], upgraded in [15] and released as a tool in [16]. Given an event stream, CDESF ingests each event at a time, produces a trace representation based on a graph structure, extracts distances related to a model, and clusters cases in the feature space. These processes are regulated by four steps: *Transformation*, *Distance Computation*, *Online Clustering* and *Check Point*. The first three steps are triggered for each new event, while the last step works as a regulator, updating the process model and releasing older data. Internally, CDESF uses a graph representation for its process model inspired by the directly-follows graph, where nodes represent activities and edges the connections between these activities. This representation supports the computation of distance metrics and makes model updates feasible. We refer to this internal representation as Process Model Graph (PMG). For more information about PMG creation and update, please refer to [15].

At the arrival of a new event, the *Transformation* phase instantiates the event as a node and produces a graph representation. When the event belongs to a case already seen, this event is added to its corresponding case representation. Otherwise, the event is the first from its case, meaning that the graph representation for this case is a single node. In earlier versions of CDESF, the node recorded only the activity name and the date and time tuple representing when the activity was executed. In this work, we extend this core representation by including all the additional attributes related to an event. Hence, the node now accumulates all the data available in the event stream. Hence, the edge represents the directly-follow relation between activities (nodes), including their frequency. Both PMG and trace representations follow this guideline to represent activities and their ordering.

Once the event has been transformed into a graph, case distances are computed in the *Distance Computation* step. These distances are based on the difference between the case graph and the PMG. Consequently, cases closer to the PMG behavior produce lower distances while cases containing less frequent behavior in the PMG result in higher distance values. In [15], case distances insisted in two perspectives: trace and time, referred to as graph-distance trace (GD_{trace}) and graph-distance time (GD_{time}). Distances are computed after normalization in the PMG. From the trace perspective, the normalization divides each node frequency by the node with the highest frequency. From the time perspective, the normalization acts on the edges by computing the mean time between two activities (nodes) occurrence. For more information on GD_{trace} and GD_{time} computation, we refer the reader to [15].

In this paper, we extend the support for multiple dimensions in online processing, meaning that graph distances are computed for the additional attributes. For numerical attributes, the distances are calculated using a similar approach as the one used to calculate GD_{time}. Given a trace tr, for each activity $tr[i]$ in the trace, we compute the difference between the value of the numerical attribute $tr[i]_{num\_attr}$ and the average value of all previous occurrences of that attribute for the same activity in the PMG. The sum of distances for the attribute across all activities is then compared to the sum of the average values in the PMG. Categorical attributes follow a similar approach as the one used to calculate GD_{trace}. Given a trace tr, for each activity $tr[i]$ in the trace and its corresponding categorical attribute's value $tr[i]_{cat\_attr}$, we compute the difference between the frequency of that value in the PMG and the number of occurrences for the same attribute. The sum of these distances is then divided by the number of nodes in the trace T_{tr}. Each event attribute has its respective distance computation. Hence, the set of distances has the same length as the number of event attributes. Fig. 1 shows a PMG and a trace already modeled as a graph. We note the attributes attached to both graphs. Considering attribute $attr_1$ and $attr_2$, $GD_{attr_1} = \frac{|\frac{10+3+8}{3}-9|+|\frac{0+6+1}{3}-5|+|\frac{2+2}{2}-2|}{7+2.33+2} = 0.4121$ and $GD_{attr_2} = \frac{(1-\frac{2}{3})+(1-0)+(1-\frac{1}{2})}{3} = 0.6111$.

We refer to the set of distances as graph distances (GD) from now on. The *Online Clustering* step uses the previously computed GD to formulate the online feature space. We adopted the DenStream [19] algorithm for online clustering. Using the concept of micro-clusters [20], DenStream dynamically creates, updates, and deletes micro-clusters from the online feature space. The density-based clustering technique incorporates the concept of three micro-cluster types: outlier micro-cluster (anomalous behavior), potential micro-cluster (potential normal behavior), and core micro-cluster (normal behavior). Denser regions are identified as common behavior as they concentrate more elements, consequently associated with core micro-clusters. Regions with low density represent anomalous behavior and are associated with outlier micro-clusters. Given new GDs, DenStream updates the current feature space by adjusting micro-clusters radius and positions. Regions that become denser as new points arrive are promoted to core micro-clusters. This way, the promotion of a new core micro-cluster illustrates the new behavior that arrived in the stream. We identify this promotion as an indicator of drift since a new behavior (previously unseen) was detected in the feature space. Note that the number of dimensions respects a one-to-one relationship with the number of event attributes.

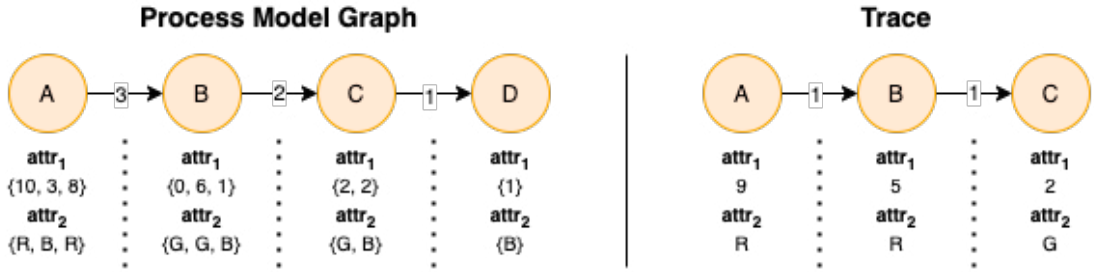

Fig. 1. An example Process Model Graph and a new trace.

The last step, *Check Point* (CP), deals with intrinsic constraints attached to stream processing, namely, model update and memory control. Since a stream is potentially infinite, regulators are necessary to control resource consumption. Moreover, due to behavior change, updating mechanisms are required to maintain the model relevant. The CP step is regulated by a hyperparameter defining the size of a time window. To control memory consumption, the CP stage releases older cases from memory according to the Nyquist sampling theorem [21], which states that a minimum threshold to the sampling rate of data ingestion is twice the highest frequency included in the signal. In our scenario, the number of unique cases that arrived after the last CP is interpreted as the highest frequency. This way, CP compares the number of cases in memory with the Nyquist threshold. If there are more cases than the threshold, then the excess of cases is released from memory. Regarding the model update, cases that arrived after the last CP (newest cases in the stream) are used to create a temporary graph, which is then merged with the PMG. Thus, the PMG incorporates both the latest behavior ingested in the stream and the behavior that is stably observed in the log. The PMG weights are decayed before merging with the temporary graph to control the balance between new and historical data. This practice decreases the importance of historical data, paving the way for the model update.

Traditionally, CDESF supported online process discovery, distance computation, and clustering based on only two dimensions (control-flow and time). In this work, we extend CDESF's capabilities by adapting the framework to handle multiple business process attributes, thus, increasing the tool coverage. More importantly, CDESF is now able to identify anomalies and drifts in the data-flow perspective, an important requirement in online environments. CDESF is available as a Python package for the community[1].

4. Experimental setup

Given CDESF's general-purpose nature, we evaluate several complimentary online tasks to leverage the understanding of the framework capabilities. This way, we propose a set of three experiments capturing different perspectives in stream processing. Each experiment aims at evaluating a singular aspect of the tool, namely: (i) drift detection (with and without data-flow attributes) in a real-life scenario, (ii) quantitative drift detection evaluation in synthetic event streams with four different drift types, and (iii) data-flow anomaly detection.

In the first experiment, we evaluate concept drift detection in real-life event data, assessing the number of drifts and their correspondent positions in the stream. A similar evaluation was performed in the literature [13] [11]. Our main goal is to compare the detected drifts with other approaches and, going further, finding drifts in additional dimensions. Thus, incorporating multiple event attributes other than only the control-flow perspective. For that, we used the *helpdesk*[2] event log as it was also used in similar works [13] [11]. The *helpdesk* log contains 4580 traces and 21348 events. To measure data-flow impact on drift detection, we used two attributes from the *helpdesk* log: *resource* and *service_level*. Both attributes are categorical, *resource* refers to the person executing the activity, containing 22 unique values, while *service_level* indicates the level at which the activity was executed, containing 4 unique values.

The second experiment quantitatively evaluates drift detection capabilities. For that, we use the synthetic event streams that are known to contain one drift and compare the results with the ProDrift tool proposed by Ostovar et al. [13]. This technique was chosen due to its proposal of processing event streams, that is, an online algorithm for concept drift detection in business processes. Other works that propose an offline detection do not meet the criteria for comparison as they process event logs as batches. The adopted performance measure for this experiment is Root Mean Squared Logarithmic Error (RMSLE) as the metric was already employed in similar works [3]. RMSLE is robust to outliers and underestimations are more penalized than overestimations, meaning that false negatives are more problematic than false positives. This is, in our opinion, in accordance with the drift detection task because not

[1] https://github.com/gbrltv/cdesf2

[2] https://doi.org/10.4121/uuid:0c60edf1-6f83-4e75-9367-4c63b3e9d5bb

detecting a drift may result in a significant drop of accuracy, while detecting an irrelevant drift means an unnecessary update of the model. To evaluate the capacity of detecting concept drifts in event streams, we used the synthetic event streams proposed in [3] and available in [22]. We selected 250 event streams containing 1000 cases each. The streams represent four drift types (sudden, recurring, incremental, and gradual) that describe the type of change that affected the process behavior. *Sudden* is an abrupt behavior change. *Recurring* marks seasonal behavioral changes. *Incremental* refers to a set of minimal and continuous changes that characterize a unique drift. *Gradual* is the type of drift where the new behavior probability increases while the older behavior probability decreases until the new behavior completely substitutes the first [24]. Moreover, all synthetic streams contain one drift in the control-flow perspective with varying percentages of anomalous instances.

In the last experiment, we evaluate CDESF's capabilities of detecting anomalies in the data-flow perspective. For that, we used one additional set of synthetic event logs containing anomalies affecting multiple perspectives proposed in [23]. We selected a subset of event logs (20 in total) with anomalies disturbing cases in the data-flow perspective, namely, the *resource* and *actor* attributes. *Resource* is an associated cost for the activity execution and *actor* is the stakeholder that executes the activity. *Resource* anomalies are related to values outside the expected range while *actor* anomalies allow forbidden users to execute an activity. For instance, given an activity A that is executed by two actors $\{E_1, E_2\}$, given a case C_1, an *actor* anomaly is injected by changing the activity executor, such as associating A with *actor* E_3. Similarly, the *resource* anomaly injects noise to the original cost of the activity. For example, given an activity B that has a cost range between $[30, 50]$ is associated with a cost of 10 for a given case C_2. In both examples, the cases contain anomalous attributes that should be identified by data-flow-aware techniques. We applied the synthetic event logs with anomalies in the data-flow attributes, thus, assessing online anomaly detection supported by CDESF.

5. Results and discussion

5.1. Real-life data

Processing the *helpdesk* event log, CDESF detected 4 drifts in the following positions: 9280, 9725, 11072, 15107. Using the same event log, the authors in [25] detected drifts in positions 8757 and 17307, whereas authors in [11] detected two drifts in the first half of the log and one in the second half. For instance, drifts in 9280 and 9725 relate to the drift detected in 8757 by [25], while drift 15107 relates to drift 17307 detected by [25]. Depending on the drift type, instead of a drift point, there is a drift region, e.g., gradual drifts are not instantaneous and affect a region. This phenomenon may explain the slight differences in positions for each tool. Moreover, CDESF detects more drifts as the other two works because it processes both control-flow and time perspectives. Thus, changes affecting the time feature space may yield drift alerts. Fig. 2 visually depicts the drift detected at position 11072. First, we observe the feature space before the drift at position 11071. Note the micro-cluster 10, a potential micro-cluster (blue ellipse) at this moment. In the right, at position 11080, the same feature space appears after more events are ingested. Additional cases were placed in this region, consequently becoming denser and promoting the micro-cluster to a core micro-cluster (black ellipse), representing a new behavior. CDESF also outputs the cases involved in this cluster, which can be analyzed for further inspection. We can observe that these cases are similar in the control-flow perspective as cases belonging to micro-clusters 0, 3, and 7, but they have a different time distribution, characterizing new behavior.

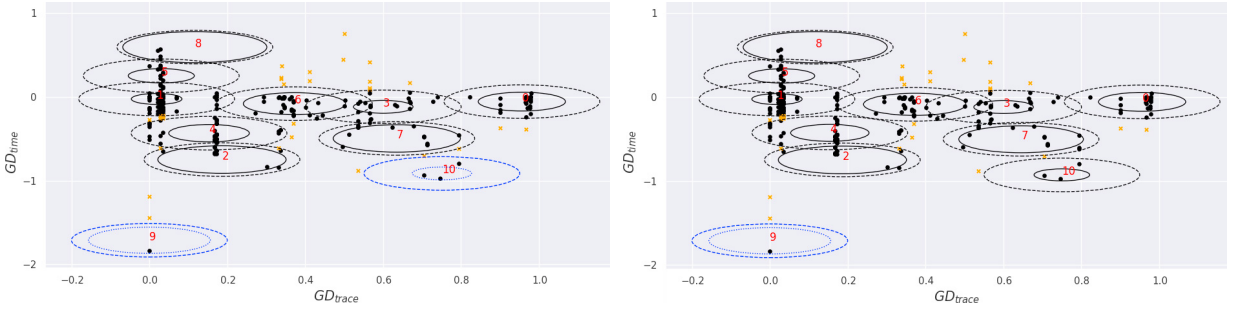

Fig. 2. Feature space in the online clustering step before (left) and after (right) a drift, located at positions 11071 and 11080. Black points are normal cases while yellow crosses represent anomalous instances. Black ellipses are core micro-clusters whereas blue ellipses are potential micro-clusters.

Furthermore, we produced a following experiment including two additional attributes: *resource* and *service_level*. CDESF detects 7 drifts when including only *resource* and 10 drifts when including only *service_level*. When including both attributes, CDESF detects 15 drifts, many of which are in the early phase of the stream. The first notable behavior is that including more dimensions may increase the number of detected drifts significantly. We can observe that

dimensions impact each other as the drift positions were not the same among the sets. Therefore, feature space distribution is directly controlled by the set of attributes arriving through the stream. Nevertheless, the results highlight the importance of incorporating the processing of additional attributes in online PM. These additional dimensions capture additional drifting behavior that, when identified, may leverage knowledge regarding a business process.

5.2. Synthetic event streams

In this experiment, we compared CDESF and ProDrift [13]. For that, 250 event logs with four drift types were streamed to the tools. Both techniques were applied with standard hyperparameters, thus, no tuning approach was employed. Because ProDrift only analyzes the control-flow perspective we selected event logs including anomalies defined on the control-flow only. This intrinsically limits the potential of CDESF. Table 1 shows the performance of both tools measured by RMSLE. When considering all the logs, ProDrift achieved 0.388 while CDESF remained with 0.694. CDESF is more sensitive to drift detection, tending to overestimate the real number of drifts, as pointed in [3]. On the other hand, ProDrift was more precise due to its statistical test approach, which is less sensitive for minor changes. Moreover, CDESF analyzes both trace and time dimensions, thus increasing the number of possible detected drifts as behavior shifts in the time feature space also triggers drift alerts. Considering each specific drift type, we note that the *sudden* drift is the easiest to be detected by both tools, with CDESF and ProDrift achieving 0.671 and 0.253, respectively. Indeed, the abrupt change of distributions highly affects encoding layers that capture the online behavior. ProDrift also performs well in detecting incremental drifts (0.353), followed by gradual (0.437) and recurring (0.454). ProDrift approach increases its performance in sudden and incremental changes, while gradual and recurring changes, which affect the behavior distribution probabilities, are less detectable by this tool. CDESF presents a good performance in gradual drifts (0.676), followed by recurring (0.714) and incremental (0.751). CDESF approach matched better behavior imposed by gradual changes as the framework slowly incorporates the drifting behavior into the model. Instead, incremental changes are less detectable as the very small scale changes are not perceived by the tool.

Table 1. Comparing CDESF and ProDrift performances for drift detection using RMSLE. Performance for each drift type is also exposed.

Technique	RMSLE_{all}	$\text{RMSLE}_{gradual}$	$\text{RMSLE}_{incremental}$	$\text{RMSLE}_{recurring}$	RMSLE_{sudden}
CDESF	0.694	0.676	0.751	0.714	0.671
ProDrift	0.388	0.437	0.353	0.454	0.253

5.3. Detecting anomalous cases based on the data-flow perspective

In this experiment, we evaluated CDESF capabilities of detecting anomalous process instances based on the data-flow perspective. For that, we identify cases as anomalous based on the case having an anomaly in the *actor* attribute, containing categorical values, the *resource* attribute, containing numerical values, or both. This means anomalies are not originated by a deviation in the control-flow but relate to anomalous values in the data attributes. The performance metrics were obtained simply by comparing the cases labeled by CDESF with the real labels from the event logs. The average accuracy for the 20 event logs was 86.1%, with 5% as standard deviation, while the average F-score was 0.92. The performance is better in event logs with a lower incidence of anomalous instances as the higher the anomaly frequency, the noisier is the stream. When too much noise is added, the PMG is negatively affected and the process representation becomes poorer. When measuring performance for each anomaly independently, we obtain an accuracy of 90.63% and an F-score of 0.9503 for the *actor* anomaly, an accuracy of 90.61% and an F-score of 0.9502 for the *resource* anomalies. The results indicate that CDESF maintains a similar performance in data-flow anomaly detection for both categorical and numerical attributes. This robustness increases the applicability of the tool as it handles different attribute types with a steady performance. We highlight that anomalous instances are not identified based on each separate dimension, instead, it considers the aggregation of all dimensions in the feature space during the online clustering step, which explains the approach robustness.

6. Conclusion

In this work, we extended a tool for drift detection in online PM. The extension enabled CDESF to incorporate online representations of additional business process attributes, i.e. the data-flow perspective. Additional dimensions should not be neglected in online processing as they may contain valuable information regarding the business process. We presented a set of three experiments capturing different necessities of online processing. In the real-life event log,

CDESF showed its capability of detecting similar drifts as other tools. Moreover, by including two data-flow attributes, we observed that more drifts are detected, hence, proving the impact of event log attributes in drift occurrences. Other drift detection techniques are uniquely attached to the control-flow, thus, ignoring all data-flow attributes, hindering their applicability. Future work should investigate the level of impact within the data attributes as different combinations lead to different sets of detected drifts. Along with this, we compared CDESF with another tool in a controlled scenario with 250 event streams containing four drift types. Finally, we investigated anomaly detection in the data-flow perspective, an aspect that should not be ignored by online PM solutions. CDESF achieves considerable performances in complementary needs of online environments, showing that solutions for specific tasks might be limited since they ignore the influence of other tasks. More importantly, we highlight that handling data-flow attributes leverages the analysis quality for online PM tools. In future work, we aim to extend the experiments, assess CDESF in different scenarios, enlarge the set of tasks CDESF offers, and improve the visualization output provided by the tool.

Acknowledgements

This article was financially supported by the program “Piano di sostegno alla ricerca 2020” funded by Università degli Studi di Milano.

References

[1] Dumas, Marlon, Wil M. P. van der Aalst, and Arthur H. M. ter Hofstede. (2005) “Process-aware information systems: bridging people and software through process technology.” *John Wiley & Sons*.

[2] Van der Aalst, Wil M. P.. (2016) “Process Mining: Data Science in Action.” *Springer, Heidelberg* **2ed**.

[3] Ceravolo, Paolo, Gabriel M. Tavares, Sylvio. Barbon Junior, and Ernesto Damiani. (2020) “Evaluation Goals for Online Process Mining: a Concept Drift Perspective.” *IEEE Transactions on Services Computing* 1–1.

[4] Gama, João. (2010) “Knowledge Discovery from Data Streams.” *Web Intelligence and Security - Advances in Data and Text Mining Techniques for Detecting and Preventing Terrorist Activities on the Web* 125–138.

[5] Hassani, Marwan, Sergio Siccha, Florian Richter, and Thomas Seidl. (2015) “Efficient Process Discovery From Event Streams Using Sequential Pattern Mining.” *IEEE Symposium Series on Computational Intelligence* 1366-1373.

[6] van Zelst, Sebastiaan J., Boudewijn F. van Dongen, and Wil M. P. van der Aalst. (2018) “Event stream-based process discovery using abstract representations.” *Knowledge and Information Systems* **54** (2): 407–435.

[7] van Zelst, Sebastiaan J., Alfredo Bolt, Marwan Hassani, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. (2017) “Online conformance checking: relating event streams to process models using prefix-alignments.” *International Journal of Data Science and Analytics* 2364–4168.

[8] Schuster, Daniel, and Sebastiaan J. van Zelst. (2020) “Online Process Monitoring Using Incremental State-Space Expansion: An Exact Algorithm.” *Business Process Management, Springer International Publishing* 147–164.

[9] Bose, R. P. Jagadeesh Chandra, Wil M. P. van Der Aalst, Indrė Žliobaitė, and Mykola Pechenizkiy. (2014) “Dealing With Concept Drifts in Process Mining.” *IEEE Transactions on Neural Networks and Learning Systems* **25** (1): 154-171.

[10] Zheng, Canbin, Lijie Wen, and Jianmin Wang. (2017) “Detecting Process Concept Drifts from Event Logs.” *On the Move to Meaningful Internet Systems. OTM 2017 Conferences* 524–542.

[11] Yeshchenko, Anton, Claudio Di Ciccio, Jan Mendling, and Artem Polyvyanyy. (2019) “Comprehensive Process Drift Detection with Visual Analytics.” *Conceptual Modeling, Springer International Publishing* 119–135.

[12] Brockhoff, Tobias, Merih Seran Uysal, and Wil M. P. van der Aalst. (2020) “Time-aware Concept Drift Detection Using the Earth Mover’s Distance.” *2nd International Conference on Process Mining* 33–40.

[13] Ostovar, Alireza, Abderrahmane Maaradji, Marcello La Rosa, Arthur H. M. ter Hofstede, and Boudewijn F. V. van Dongen. (2016) “Detecting Drift from Event Streams of Unpredictable Business Processes.” *Conceptual Modeling, Springer International Publishing* 330–346.

[14] Barbon Junior, Sylvio, Gabriel M. Tavares, Victor G. Turrisi da Costa, Paolo Ceravolo, and Ernesto Damiani. (2018) “A Framework for Human-in-the-Loop Monitoring of Concept-Drift Detection in Event Log Stream.” *Companion Proceedings of the The Web Conference 2018* 319—326.

[15] Tavares, Gabriel Marques, Paolo Ceravolo, Victor G. Turrisi Da Costa, Ernesto Damiani, and Sylvio Barbon Junior. (2019) “Overlapping Analytic Stages in Online Process Mining.” *IEEE International Conference on Services Computing (SCC)* 167–175.

[16] Mora, Davide, Paolo Ceravolo, Ernesto Damiani, Gabriel M. Tavares. (2020) “The CDESF Toolkit: An Introduction.” [Online]. Available: `http://ceur-ws.org/Vol-2703/paperTD8.pdf`.

[17] Krawczyk, Bartosz, Leandro L. Minku, João Gama, Jerzy Stefanowski, and Michał Woźniak. (2017) “Ensemble learning for data stream analysis: A survey.” *Information Fusion* **37**: 132–156.

[18] De Medeiros, AK Alves, Boudewijn F. van Dongen, Wil MP Van der Aalst, and A. J. M. M. Weijters. (2004) “Process mining: extending the alpha-algorithm to mine short loops.” *Technische Universiteit Eindhoven.*

[19] Cao, Feng, Martin Estert, Weining Qian, and Aoying Zhou. (2006) “Density-based clustering over an evolving data stream with noise.” *Proceedings of the International Conference on Data Mining* 328–339.

[20] Aggarwal, Charu C., S. Yu Philip, Jiawei Han, and Jianyong Wang. (2003) “A framework for clustering evolving data streams” *Proceedings of the 29th International Conference on Very Large Data Bases* 81–92.

[21] Landau H. J. Sampling. (1967) "Sampling, data transmission, and the Nyquist rate." *Proceedings of the IEEE* **55** (10): 1701–1706.
[22] Tavares, Gabriel Marques, Sylvio Barbon Junior, and Paolo Ceravolo. (2019) "Synthetic Event Streams." *IEEE Dataport* [Online]. Available: https://dx.doi.org/10.21227/2kxd-m509.
[23] Barbon Junior, Sylvio, Paolo Ceravolo, Ernesto Damiani, and Gabriel M. Tavares. (2021) "Evaluating Trace Encoding Methods in Process Mining." *From Data to Models and Back, Springer International Publishing* 174–189.
[24] Gama, João, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. (2014) "A Survey on Concept Drift Adaptation." *ACM Comput. Surv.* **46** (4): 44:1–44:37.
[25] Ostovar, Alireza, Sander JJ Leemans, and Marcello La Rosa. (2020) "Robust Drift Characterization from Event Streams of Business Processes." *ACM Trans. Knowl. Discov. Data* **14** (3): 30.