The CDESF Toolkit: An Introduction

Davide Mora*, Paolo Ceravolo*, Ernesto Damiani†, Gabriel Marques Tavares*
*Università degli Studi di Milano (UNIMI), Milan, Italy
Email: davide.mora@studenti.unimi.it, {paolo.ceravolo, gabriel.tavares}@unimi.it
†Khalifa University (KUST), Abu Dhabi, UAE
Email: ernesto.damiani@kustar.ac.ae

***Abstract*—Real-time response is crucial in many business process scenarios, however, few tools support the online processing of Process Mining tasks. In this paper, we present Concept Drift in Event Stream Framework (CDESF), a tool focused on concept drift detection that also supports several online Process Mining tasks. CDESF highlights the process model evolution during the stream processing and alerts the detection of new drifts aided by an online clustering layer. This paper presents CDESF as a tool that is available for the community and process practitioners.**

***Index Terms*—Online process mining, clustering, concept drift detection, process model graph**

I. Introduction

Process Mining (PM) aims at extracting information from event logs, which are recorded by information systems during the execution of business processes. Further, PM provides insights from recorded business processes, such as the creation of a process model, detection of anomalous executions, and extraction of process metrics [1]. A relevant aspect in traditional PM is the offline assessment of event logs. That is, the analyzed event logs depict past process executions, which have already finished. However, on many occasions, stakeholders are interested in understanding the current state of the process, i.e., while the process is being performed. Many real applications cannot wait for a complete process cycle to take action, which leverages the necessity for online processing of event logs. This way, online PM has emerged as an area with the goal of handling event streams, that is, events are processed as their generation occurs.

In offline settings, PM tasks can be performed asynchronously, i.e., one might first discover a process model, then apply a conformance checking technique to detect deviations and extract metrics, and so on. However, online PM imposes constraints that make traditional processing unfeasible. Namely, like in stream mining, online scenarios must assume that the flow of events is continuous, fast, and possibly infinite. This poses the first constraint, the limitation of time and memory. Algorithms are required to have forgetting mechanisms to keep memory consumption viable (as it is impossible to store an infinite stream) and low time consumption (as the processing of events should be faster than the arrival rate of events). Moreover, PM practiced in online scenarios must also consider concept drifts, a phenomenon characterized by the change of the relation between a feature vector and its associated class over time, and also consider incomplete traces. Lastly, an updated version of the process model is needed to assess the differences between reference and real executions correctly. Therefore, methods should be aware of the process model and detect deviations at the same time while maintaining an updated model and being prepared to handle concept drifts [2].

Several methods for online PM have been proposed in the last few years, ranging from online process discovery [3]–[5], to conformance checking [6]–[8] and concept drift detection [9]–[11], but they usually tackle only one of the required needs in online environments. Moreover, though many works provide theoretical and practical advancements, at the same time, they do not provide frameworks and tools for intermediate or end-user. To overcome the reported issues, we propose the use of Concept Drift in Event Stream Framework (CDESF), a tool for the monitoring of concept drift in online process mining. The formal background of the tool was proposed in [12] and advanced in [13]. We developed a framework with the main characteristics being:

- Online process discovery: a process model graph (PMG) is maintained online and updated using the most recent events. The updating step keeps the model relevance while it serves as the representation of the business process behavior.
- Online conformance checking: the framework models both the process model and the traces as graphs. The decision is based on the broad support of graph operations. This way, metrics are extracted by comparing trace and process model. Further, the obtained metrics allows the detection of deviations from normal behavior.
- Online clustering: the tool is equipped with an online clustering technique. The clustering step places cases in the feature space, identifying regions of interest, such as common and anomalous behavior.
- Concept drift detection: the clustering step is based on the concept of micro-clusters. Given the micro-clusters behavior over time, a concept drift is detected.

To support the listed characteristics, CDESF follows two main guidelines:

- Memory consumption: a forgetting mechanism controls the release of old cases from memory, thus, handling constraints posed by online scenarios.
- Support to event stream processing: the approach handles

This study was also partly supported by the program "Piano di sostegno alla ricerca 2019" funded by Università degli Studi di Milano.

event streams (not to be mistaken with trace streams) ingesting one event at a time unit.

CDESF tackles several problems faced in online PM and provides in-depth insights into the business process for stakeholders. This way, moving forward the current state of tools available for users and PM enthusiasts. Recently, CDESF has been released as a Python library for the community[1]. A video demonstrating the framework's use and its capabilities is available[2]. Moreover, a tutorial showing detailed information on how to use the framework, along with a deeper discussion of its visualizations is also available[3]. We also note that CDESF has been used in a pipeline with PM4Py [14] in previous works, an integration that provides further support for online PM tasks.

The remaining of this paper is organized as follows: Section II presents CDESF architecture and its main features. Then, Section III shows the tool maturity, including several supported analysis by giving some examples. Finally, Section IV leaves the concluding remarks.

II. CDESF Architecture

Figure 1 exposes CDESF architecture and how each step is performed. Notice that the event stream is possibly infinite and that events from different cases are interspersed. This way, at the arrival of a new event, its case must be retrieved and complemented with the event. For each new event, CDESF steps 1 to 3 are processed, i.e., the framework does not consume data in chunks, such as window-based techniques.

When a new event arrives, the *Transformation* step adds the event into its corresponding case and models the case into a graph-based representation. If the new event belongs to a new case (i.e. a case that has never been seen before), the case is initialized with the event. Then, the graph representation will contain a single event. Instead, if the event is from an already existing case, the case is updated with the event. Consequently, the graph representation is also updated by adding a new node. Graph nodes and edges represent events and directly-follow relations, respectively.

The second step, *Distance Computation*, aims to extract features that describe the case behavior. For that, the case is compared with the PMG, which captures the current business process nature. The comparison catches two perspectives: *trace* and *time*, noted as graph-distance trace (GD_{trace}) and graph-distance time (GD_{time}). Distances are measured in a normalized PMG. From a trace perspective, the normalization is the occurrence of a specific transition divided by the value of the most occurred transition. The result of this operation is referred to as *weight*. For the time perspective, edge normalization computes the mean time of a transition between activities. Equations 1 and 2 show the computing of GD_{trace} and GD_{time}, respectively.

$$GD_{trace}(tr) = \frac{\sum_{i=1}^{T_{tr}} 1 - PMG_{weight}(tr[i])}{T_{tr}} \quad (1)$$

[1]https://github.com/gbrltv/cdesf2
[2]https://youtu.be/Hq3xLyZOmlg
[3]https://github.com/gbrltv/cdesf2/blob/master/tutorial.pdf

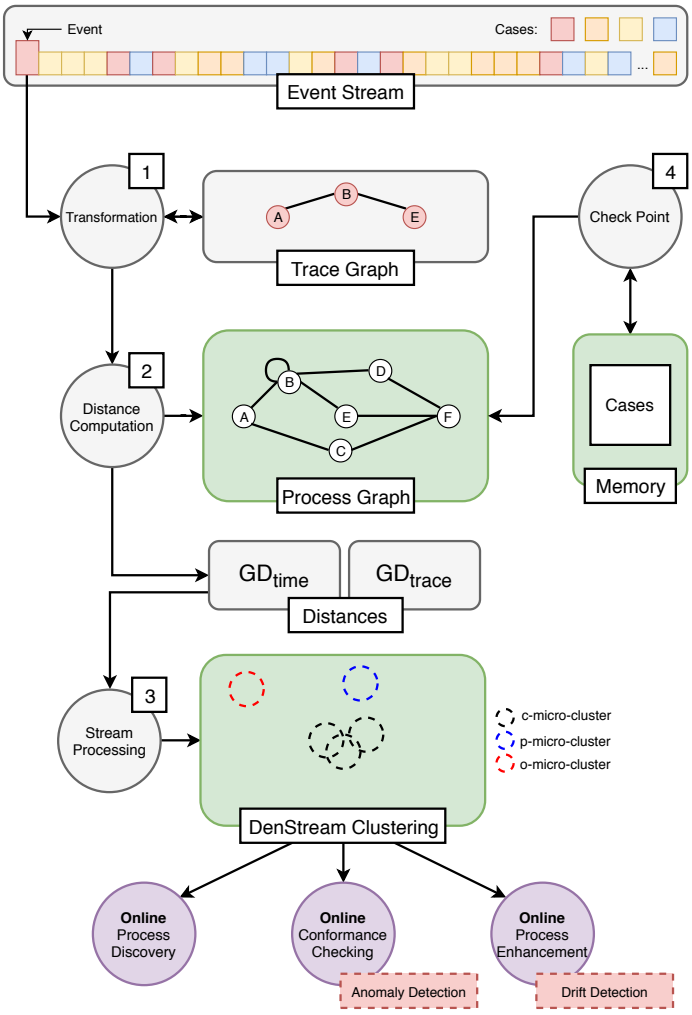


Fig. 1: CDESF architecture is divided into four main steps. First, the event case is retrieved and transformed into a graph. Then, distances between the trace and process model are extracted. These metrics are fed to a density-based online clustering for anomaly and concept drift detection. Finally, a seasonal step updates the model and releases inactive cases from memory.

$$GD_{time}(tr) = \log_{10}\left(\frac{\sum_{i=1}^{T_{tr}} |PMG_{time}(tr[i]) - tr[i]_{time}|}{\sum_{j=1}^{T_{tr}} PMG_{time}(tr[j])}\right) \quad (2)$$

Given a trace tr, T_{tr} is the total amount of edges in tr, $tr[i]$ corresponds to the ith edge in tr, $tr[i]_{time}$ is the time delta in $tr[i]$ and $tr[i]_{weight}$ is the weight of $tr[i]$. The graph distances capture two essential aspects of traces, the control-flow and the time between activities. And given the flexibility of the framework, it is possible to add multiple dimensions considering additional perspectives, e.g., event attributes.

The graph distances previously calculated (GD_{trace} and GD_{time}) are fed to a density-based clustering algorithm (DenStream [15]), which identifies common behavior over time and highlights outliers. DenStream provides an aggregation of similar process behavior, where denser regions represent

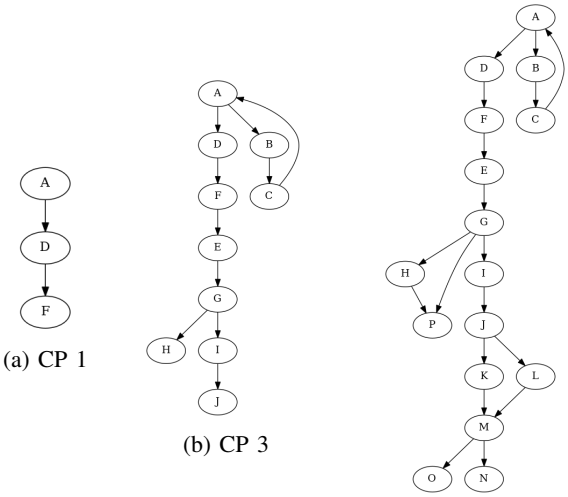
common patterns and sparser regions represent anomalous patterns. Moreover, it detects cases that do not belong to any cluster or are not dense enough to form one. These points are also considered anomalous since they are isolated in the feature space. DenStream uses the concept of micro-clusters and further classifies micro-clusters in three types: outlier micro-cluster (anomalies), potential micro-cluster (potentially common behavior), and core micro-cluster (common behavior). The detection of concept drift relies on the detection of new core micro-clusters, i.e., if a new behavior has appeared in the stream and is dense enough to form a core micro-cluster, then a drift has been detected.

The *Check Point* (CP) step has two main goals: memory control and model update. It is controlled by a hyperparameter (in time unit) that sets a time horizon to how frequent CPs happen. Regarding model update, all new cases from the last time horizon are used to build a temporary graph. The temporary graph represents the newest behavior of the process since it is built with the newest events. Then, the temporary graph is merged into the PMG to update the process representation. Before merging, a decay factor is applied to the PMG to balance new and historical data. If a directly-follows relation stops appearing, its weight decreases gradually. This step supports process model enhancement by using new information to maintain the PMG faithful to real data. Regarding memory control, older cases are released from memory according to the Nyquist sampling theorem [16]. The use of Nyquist facilitates the specialist role, taking away the need for manual setting for a forgetting mechanism.

III. Tool Maturity

As an algorithm, CDESF has been used in several scientific researches [7], [12], [13], [17]. It was also tested using 942 event logs with several types of concept drift and anomalous behavior [18] obtaining great performance [2].

One of the main interests in PM analysis is to evaluate the process model, i.e., process discovery. Traditionally, given a complete event log, the relations between events are inferred to build a process model. In a stream of events, algorithms have to periodically update the process model since recent events may represent new behavior. This way, a constant model update component is required, so the process always represents the newest behavior seen in the stream. In CDESF, the CP step controls model updates using the most recent events that have arrived in the last time window. Figure 2 shows the PMG in different stages of the stream. In the first CP (Figure 2a), the PMG is still very simple given the low amount of events that have arrived. Note that CDESF does not start stream processing with a pre-built model, instead, it learns the model during the stream processing. This way, it is expected that during the first CPs, the process model is simple since the presented behavior of the stream is minimal. Already on CP 3 (Figure 2b), we can observe how the PMG has embedded more information. And on CP 28 (Figure 2c), an even more complete PMG has been created. There is a clear relationship between the time horizon hyperparameter, which controls CP frequency, and the PMG update. Higher time horizons imply in a longer time between updates, this is advised in processes where more constant behavior is expected. Contrarily, lower time horizons have more frequent CPs triggering more updates, which is advised for faster streams or processes where change is expected. Other than the visual output, CDESF also saves the PMGs during every CP in a JSON format, increasing the possibilities of use in different software.



(a) CP 1 (b) CP 3 (c) CP 28

Fig. 2: Evolution of the PMG through several check points (CP). In the first CP, the PMG is still very basic since very few events have arrived in the stream. With the consumption of more events over time, the PMG is updated to represent new behavior better, as seen in CPs 3 and 28.

Regarding drift alerts, CDESF has two main responses. During the stream processing, CDESF triggers drift alerts when a drift is detected. This approach facilitates decision making for organizations using the framework since the drift is detected and alerted on the fly. Moreover, CDESF also creates a visual output after stream processing showing the number of detected drifts and their positions. This visualization is shown in Figure 3. Vertical dashed lines represent the point in time where drifts were detected. At the same time, the figure demonstrates the cumulative number of drifts detected in the stream. In this scenario, further behavioral analysis is possible. All the seven drifts were detected before half of the stream. This means that the process suffered many changes in the initial stages, and with time, the process execution got a stabler form.

CDESF is also capable of representing the evolution of clusters in the stream. Figure 4 shows the clustering feature space after 2950 processed events. Core, potential, and outlier micro-clusters are represented in black, blue and red, respectively. The micro-clusters contain two rings, the dashed line is the maximum range a micro-cluster can achieve (controlled by a hyperparameter), while the continuous line is the actual radius of the micro-cluster. It follows that the points (cases) are

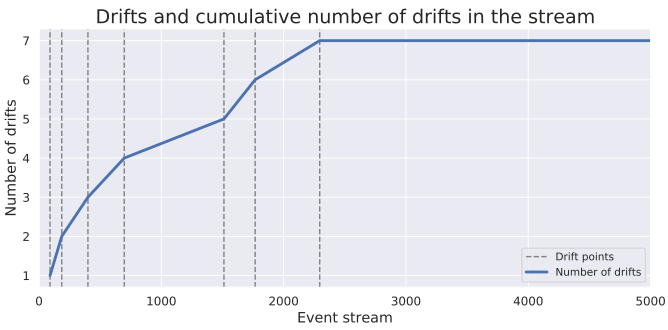


Fig. 3: Relation of the event stream processing, number of detected drifts and their positions. Vertical lines represent the drift positions and the curve shows the cumulative number of drifts.

represented in two ways. The black dot is a normal case falling under a core micro-cluster area, while the yellow marker represents anomalous behavior, i.e., cases that do not conform to the PMG.

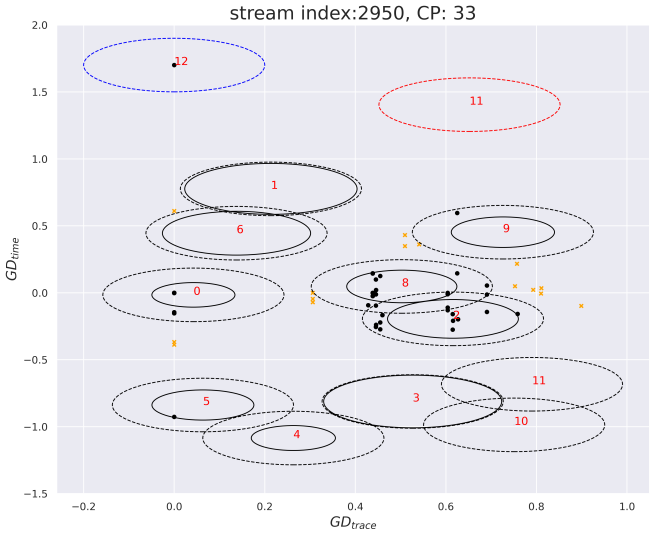


Fig. 4: Feature space after 2950 events processed. Core micro-clusters (black regions) represent common process behavior. Normal and anomalous cases are shown in black dots and yellow markers, respectively.

All visualizations provided by CDESF are complemented by the metrics which are saved after the processing. CDESF provides the evolution of clusters and case metrics, which opens opportunities for further analysis. Both visualizations and metrics information can be crossed in many ways to offer in-depth insights into the business process.

IV. Conclusion

In this paper, we present CDESF, a tool for detection of concept drifts and monitoring of business processes in online scenarios. CDESF handles conflicting goals (memory consumption and accuracy) in stream processing, maintains a process model and supports drift detection using an online clustering layer. CDESF is an available open-source tool implemented as a library in Python.

References

[1] W. M. P. van der Aalst, *Process Mining: Data Science in Action*, 2nd ed. Heidelberg: Springer, 2016.

[2] P. Ceravolo, G. M. Tavares, S. Barbon, and E. Damiani, "Evaluation goals for online process mining: a concept drift perspective," *IEEE Transactions on Services Computing*, pp. 1–1, 2020.

[3] S. J. van Zelst, B. F. van Dongen, and W. M. van der Aalst, "Event stream-based process discovery using abstract representations," *Knowledge and Information Systems*, vol. 54, no. 2, pp. 407–435, 2018.

[4] V. Leno, A. Armas-Cervantes, M. Dumas, M. La Rosa, and F. M. Maggi, "Discovering process maps from event streams," in *Proceedings of the 2018 International Conference on Software and System Process*, ser. ICSSP '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 86–95.

[5] A. Burattin, A. Sperduti, and W. M. van der Aalst, "Control-flow discovery from event streams," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*. IEEE, 2014, pp. 2420–2427.

[6] A. Burattin and J. Carmona, "A framework for online conformance checking," in *Business Process Management Workshops*, E. Teniente and M. Weidlich, Eds. Cham: Springer International Publishing, 2018, pp. 165–177.

[7] G. M. Tavares, V. G. T. da Costa, V. E. Martins, P. Ceravolo, and S. Barbon, Jr., "Anomaly detection in business process based on data stream mining," in *Proceedings of the XIV Brazilian Symposium on Information Systems*, ser. SBSI'18. New York, NY, USA: ACM, 2018, pp. 16:1–16:8. [Online]. Available: http://doi.acm.org/10.1145/3229345.3229362

[8] P. Koenig, J. Mangler, and S. Rinderle-Ma, "Compliance monitoring on process event streams from multiple sources," in *2019 International Conference on Process Mining (ICPM)*, 2019, pp. 113–120.

[9] R. P. J. C. Bose, W. M. P. van der Aalst, I. Žliobaitė, and M. Pechenizkiy, "Handling concept drift in process mining," in *Advanced Information Systems Engineering*, H. Mouratidis and C. Rolland, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 391–405.

[10] A. Yeshchenko, C. Di Ciccio, J. Mendling, and A. Polyvyanyy, "Comprehensive process drift detection with visual analytics," in *Conceptual Modeling*, A. H. F. Laender, B. Pernici, E.-P. Lim, and J. P. M. de Oliveira, Eds. Cham: Springer International Publishing, 2019, pp. 119–135.

[11] A. Ostovar, S. J. Leemans, and M. L. Rosa, "Robust drift characterization from event streams of business processes," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 14, no. 3, pp. 1–57, 2020.

[12] S. Barbon Junior, G. M. Tavares, V. G. T. da Costa, P. Ceravolo, and E. Damiani, "A framework for human-in-the-loop monitoring of concept-drift detection in event log stream," in *Companion Proceedings of the The Web Conference 2018*, ser. WWW '18. International World Wide Web Conferences Steering Committee, 2018, pp. 319–326.

[13] G. M. Tavares, S. Barbon Junior, P. Ceravolo, and E. Damiani, "Overlapping analytic stages in online process mining," in *2019 IEEE International Conference on Service Computing (SCC 2019)*, July 2019.

[14] A. Berti, S. J. van Zelst, and W. van der Aalst, "Process mining for python (pm4py): Bridging the gap between process and data science," 2019.

[15] F. Cao, M. Estert, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in *Proceedings of the 2006 SIAM international conference on data mining*. SIAM, 2006, pp. 328–339.

[16] L. Lévesque, "Nyquist sampling theorem: understanding the illusion of a spinning wheel captured with a video camera," *Physics Education*, vol. 49, no. 6, pp. 697–705, 2014.

[17] N. J. Omori, G. M. Tavares, P. Ceravolo, and S. Barbon, "Comparing concept drift detection with process mining software," *iSys - Revista Brasileira de Sistemas de Informação*, vol. 13, no. 4, pp. 101–125, 2020.

[18] G. M. Tavares, S. Barbon, and P. Ceravolo, "Synthetic event streams," 2019. [Online]. Available: http://dx.doi.org/10.21227/2kxd-m509