Adaptive Interactive Process Drift Detection: Detecting and Visualizing Process Drifts

Denise Maria Vecino Sato[1][0000−0003−1117−7082], Sheila Cristiana de Freitas[1][0000−0003−3688−4066], Jefferson Koji Sato[2][0009−0007−4297−6529], Jean Paul Barddal[2][0000−0001−9928−854X], and Edson Emilio Scalabrin[2][0000−0002−3918−1799]

[1] Instituto Federal do Paraná Campus Curitiba, João Negrão, 1285, Curitiba, Brazil
{denise.sato,sheila.freitas}@ifpr.edu.br
[2] Programa de Pós-Graduação em Informática (PPGIa) da Pontifícia Universidade Católica do Paraná, Imaculada Conceição, 1155, Curitiba, Brazil
{jefferson.sato,jean.barddal,scalabrin}@ppgia.pucpr.br

Abstract. Process mining extracts insights about business processes from information system data. However, traditional techniques often assume static processes, which is unrealistic. Detecting process drifts is crucial for accurate analysis, but existing methods lack consistent detection due to parameter sensitivity and a lack of a standard comparison protocol. This paper introduces the Adaptive Interactive Process Drift Detection (IPDD), which applies the ADWIN change detector to process model quality metrics over time. Adaptive IPDD continuously assesses fitness and precision metrics to detect drifts. Results show IPDD's effectiveness in synthetic datasets, comparable to Apromore in drift detection and outperforming Apromore AWIN in Mean delay. We also highlight that IPDD exhibits stable performance even with low window size values. We also evaluate a real-life event log representing an Italian company's ticketing management process using Adaptive IPDD. The results demonstrated that the drift analysis for real scenarios can be improved by exploring the user interface of IPDD.

Keywords: Process Drift Detection · IPDD · Process Mining · ADWIN

1 Introduction

Process mining aims to obtain valuable knowledge from business processes based on actual data collected from the information systems. The event data must contain at least the performed activity (a step in the process), the case identifier (to identify the process instance), and the timestamp, usually stored as an event log. Each process execution for a specific case generates a trace in the event log, representing the sequence of activities performed for the case in the analyzed business process. The most common tasks in process mining are discovery, conformance, and enhancement. Using a discovery algorithm, we can obtain a process model from the event log without any *a priori* information. In conformance checking, we compare the event log with a process model (designed or

discovered) to understand deviations in the behavior of the process executions. Moreover, we can enhance the discovered model by including additional perspectives from the event log, e.g., resources and performance [2].

A challenging trait of processes is their evolving nature, i.e., processes change due to new regulations and dealing with urgent or other situations. Therefore, detecting these changes in process behavior may raise relevant information for the business analysts, e.g., when the process was changed, what differences are in the distinct versions of the process, and what is the process's current version. The situation where the process changes while being analyzed is called *concept drift* or *process drift* [2]. The authors in [5] define three challenges for dealing with drifts: (i) change point detection – define the point in time where the change occurred; (ii) change localization and characterization – localize a model change and characterizing it, and; (iii) change process discovery – discover the changing process by considering the previously identified information.

A process usually describes the control-flow perspective, i.e., the sequence of activities allowed in the process. A process drift in this perspective might represent a structural change in the model, e.g., adding or removing an activity or changing two activities from a sequential to a parallel structure. Also, it may represent a change in the behavior of the process, i.e., in the routing of cases [5]. For our approach, we only detected the structural changes in the process.

The changes in the processes can affect the current instances suddenly or gradually, named as sudden or gradual drift in [5]. In a sudden drift, when a new version of the process starts, it affects all the current instances; for example, a pandemic situation will immediately change the healthcare protocols for all patients. However, in a gradual drift, the new instances follow the new process, and existing ones follow the former version, e.g., a new loan assessment rule must be applied to the new customers. Furthermore, versions of the processes can reappear, or the latest version can be implemented using small changes over time, generating recurrent and incremental drifts as defined by [5]. The process drifts may be analyzed in both offline and online fashions. An offline analysis requires an event log as input and may use the drift information to understand, design, or improve processes. On the other hand, online analyses are more suitable when the changes need to be discovered in near real-time so that such changes can also instigate online model adaptations.

This paper presents a new process drift detection tool for offline analysis that addresses change point detection, localization, and change process discovery for the sudden control-flow process drifts. The tool extends the Interactive Process Drift Detection (IPDD) framework using an adaptive windowing approach named Adaptive IPDD. Besides the new approach, IPDD provides a web interface fully available online for identifying and investigating drifts in event logs[3]. IPDD's source code and the synthetic datasets applied for validation are also available, allowing massive drift analysis or more complex scenarios[4].

[3] https://visual-pro-drift.com.br/

[4] https://github.com/denisesato/InteractiveProcessDriftDetectionFW - branch: update_libraries

2 Related Works

The survey in [16] described the current approaches for dealing with process drifts, highlighting that most approaches focus on offline drift detection, usually detecting the changing points for sudden drifts in the control-flow perspective. However, drift detection still requires efforts towards different challenges, such as the lack of a protocol for comparing results between tools and the impact of the parameters on detection accuracy.

The Apromore ProDrift [10, 9, 13, 12] implements a statistically grounded approach for detecting process drifts in event logs (runs approach) and streams (events approach). Both approaches apply statistical hypothesis testing over the distributions of runs or alpha relations (events approach) observed in adjacent time windows. The plugin provides an adaptive approach for addressing the challenge of defining the window size and two methods for change characterization and localization [12]. The main challenge in Apromore is to define a suitable initial window size. Furthermore, the user interface does not allow visualizing the different versions of the process model and does not provide evaluation metrics.

The Visual Drift Detection (VDD) [17] detects sudden, gradual, recurring, and incremental drifts and provides plots for visual analysis of the drift characterization and localization. VDD applies a change point detection algorithm (PELT) in a multivariate time series containing pre-defined measures calculated over *Declare constraints* [1], derived from the event log. The user can combine a clustering strategy to identify local behavior changes. The drift is indicated as a constraint in the process map, and the user may have to determine the change point by inspecting other visualizations, e.g., by checking each cluster's Drift Chart. VDD requires three parameters: window size, window step, and cut threshold (for the clustering strategy). One drawback of this method is that the detected change points are sensitive to the parameter configuration. When dealing with synthetic datasets, neither Apromore ProDrift nor VDD reports the evaluation metrics, e.g., F-score and Mean delay on the user interface.

The Fixed IPDD [14], based on the IPDD Framework, aims to overcome the following issues: a not-so-user-friendly interface, the difficulty in comparing results obtained by different parameter configurations, complex configuration, and not commonly reporting the accuracy metrics. The Fixed IPDD applies a fixed window approach that splits the event log based on the user-given window parameter (w). Then, it mines a Directly-Follows Graph (DFG) for each window and compares the adjacent DFGFs using two metrics: the similarity between the nodes and edges. Whether one of the metrics reports a value less than one, the drift is reported. The Fixed IPDD was validated using a public, synthetic dataset containing 18 change patterns. The definition of the w parameter is still challenging. However, the Fixed IPDD provides an interactive and easy user interface, freeing the user to test different parameters quickly and helping the process define the window parameter. The interface also provides the F-score metric. Besides the contribution of the tool, their application in real scenarios still faces challenges because of the dependency on the w parameter.

The C2D2 [6] is a conformance-based approach for detecting sudden drifts in the control-flow perspective. The authors analyze fitness and precision to identify process model changes. The authors propose two new metrics with low computational costs instead of using the fitness and precision metrics available in process mining tools, e.g., PM4Py [3]. Based on the minimum window size, C2D2 adjusts a sliding window as input for discovering a model and calculating the metrics. Then, it applies simple linear regression using the last $n/2$ conformance measures and a statistical test over the regression slope: if the slope is 0, the window is a drift candidate. Then, the algorithm checks if the reported drift persists over time by analyzing the last traces.

Adaptive IPDD differs from C2D2 because it applies the ADWIN detector [4] to identify the drifts. Also, the experimental protocol applies only one configuration for each compared tool. We propose an experimental protocol for evaluating drift detection methods, including new synthetic datasets, setting up experiments using a range of configuration parameters, selecting evaluation metrics, and reporting results. This protocol allows researchers to compare methods more efficiently and ensure that comparisons are consistent and fair.

The Adaptive IPDD detects sudden drifts in the control-flow perspective by evaluating the fitness and precision metrics over time and applying the ADWIN detector [4]. We compare it to the Apromore ProDrift [10, 9, 13, 12], VDD [17] and Fixed IPDD [14]. Furthermore, IPDD calculates and reports two evaluation metrics (F-score and Mean delay) and provides a web interface for localizing and visualizing the changing process over time. We plan to include more recent tools, e.g., C2D2 [6] in the experimental protocol as future work.

3 Adaptive IPDD for Control-Flow Drifts

A discovered process model represents the behavior of the process obtained from the event log. The quality of the model is characterized by four dimensions: fitness, precision, generalization, and simplicity [2]. The fitness dimension measures how much of the behavior in the log is allowed by the model. As the event log represents a snapshot of the complete process, the generalization dimension indicates that the model should generalize these sampled behaviors. However, the model should not allow behavior utterly unrelated to the one observed in the log, measured by the precision dimension. Finally, the simplicity dimension indicates that the model should be as simple as possible. The available discovery algorithms pursue a trade-off between these four quality dimensions [2].

After collecting a minimal number of traces, it is possible to apply a discovery algorithm to obtain a process model and then evaluate some quality metrics, e.g., fitness and precision. In a steady-state situation, i.e., the process does not change, we can re-evaluate the same quality metrics after collecting more traces, and they should stay stable. However, the traces represent a different behavior after the change point in a process drift situation. Therefore, the values for the same quality metrics evaluated using the previously discovered process model and the new traces (after the drift) will change. Adaptive IPDD identifies when

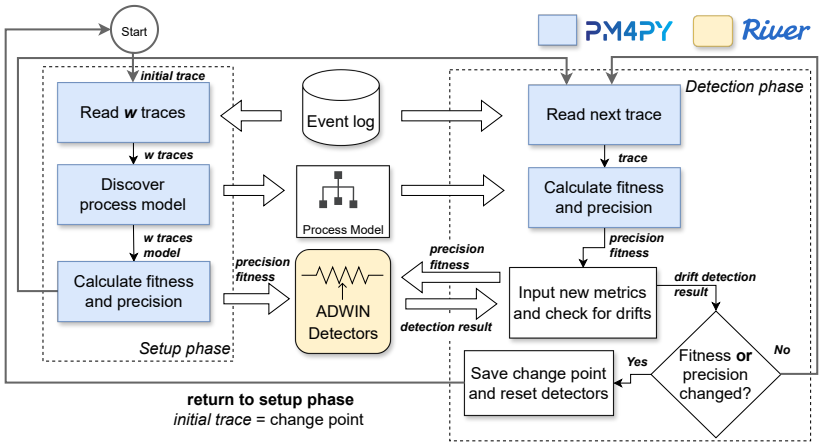


Fig. 1. Adaptive windowing strategy using conformance metrics and ADWIN.

the process changes by applying a change detector algorithm to the calculated fitness and precision metrics.

The Adaptive IPDD extends the IPDD Framework architecture [14] for detecting process drifts in the control-flow perspective using adaptive windows. The motivation is to minimize the impact of the parameters on the detection accuracy by adapting the generated window sizes. We implemented a new approach in the *Windowing strategy* module, described in Fig. 1, which cuts the windows on the detected fitness or precision changes reported by the ADWIN change detector [4]. We apply the ADWIN change detector because it has an upper on false positives and negatives and is parameter assumption-free. We use the implementation of the ADWIN change detector from the River library [11].

To execute the Adaptive IPDD, the user must set the window size w, which indicates the number of traces the method should consider to discover the process models (step *Discover process model* in Fig.1). We assume the quality metrics stay stable if the process model does not change over time. Based on this assumption, the new *Windowing strategy* contains two phases: setup and detection. In the setup phase, the method discovers an initial process model, using the w initial traces and the Inductive Miner implementation from PM4Py [3], a commonly used algorithm that discovers a sound, fitting, block-structured model from event logs in a finite time [8]. Then, it calculates the fitness and precision metrics[5] for the w initial traces and provides the calculated values to the ADWIN detectors - one for each metric. The method reads the following trace in the detection phase and calculates the fitness and precision metrics using the initial model discovered (mined during the setup phase). Next, it updates the ADWIN detectors with the new values and checks for a change. If a drift is detected for any metric, the Adaptive IPDD saves the change point and returns

[5] Calculated using token-replay (PM4Py) because they are less computationally expensive than the alignments-based metrics. Both metrics are multiplied by 100.

to the setup phase, discovering a new model with the following w traces. If no drift is detected, the method continues on the detection phase by reading the following trace in the event log. The detected change points are used to split the event log in windows, which are reported to the next step of the framework.

The *Process discovery* and *Model-to-Model comparison* steps stay with the exact implementation as described in [14, 15]. In the *Process discovery*, IPDD applies the DFG miner discovering process maps from the traces inside the provided windows using the PM4Py [3]. The adjacent derived process maps are compared in the *Model-to-Model comparison* module using the Nodes (NS) and Edges similarity (ES) metrics described in [14]. So, even if the ADWIN detector identifies the change in the quality metrics, Adaptive IPDD verifies if the DFGs of the adjacent derived windows are different in terms of nodes or edges. The user can also configure it if he is only interested in changes related to the activities of the process, for instance, by selecting only the node's similarity metric.

We enhanced the *Evaluation* module proposed in [14], reviewing the F-score and including the Mean delay metric. The F-score is the harmonic mean between precision and recall, which relies on true positives (TP), false positives (FP), and false negatives (FN). A TP is a detected drift related to a real drift (only the first detected drift after a real one[6]); an FP is a detected drift unrelated to a real drift, and an FN is a real drift not detected. The F-score results in a value between 0 and 1, measuring the accuracy of the drift detection. However, the F-score does not measure if the reported change point is "close" to the actual change point. The Mean delay complements the analysis by measuring the distance between the detected and real change points.

Fig.2 shows a possible detection scenario using a synthetic log. In the x-axis, we can see the traces ordered by timestamp $(t_1, t_2, \ldots, t_n)$. The event log contains three real drifts at t_3, t_6, and t_{20}. The method also detects 3 drifts at t_{12}, t_{24}, and t_{37}. The F-score will consider a TP a drift detected after a real drift and before the following known drift – t_{12} and t_{24}. The distance (d) between the detected drift (CP_d) and the real one (CP_r) is the number of traces between them. In Fig.2, $TP = 2, FP = 1, FN = 1$, resulting in a F-score=0.67 and Mean delay = 5, c.f. Equations 1 and 2.

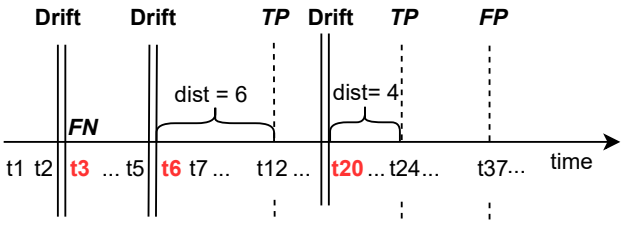


Fig. 2. Possible synthetic scenario with three real and three detected drifts.

[6] The C2D2 paper [6] considered a TP a detected drift in a neighborhood of the actual drift, even when the detected drift is before the actual one, which is different from our proposal.

$$\text{F-score} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{1}$$

$$\text{Mean delay} = \frac{\sum_{i=0}^{TP} d(CP_d, CP_r)}{TP} \tag{2}$$

The user visualizes the evolution of the process by checking the model for each window. After selecting a window, the user can localize the change by checking the similarity metrics information in the web interface[7] [14, 15]. If the user executes Adaptive IPDD via the command line or the massive interface[8], they can check the similarity information provided by the outputted files.

4 Experimental Setup

We validated Adaptive IPDD using two synthetic datasets (dataset 1 and dataset 2), respectively containing 68 and 52 event logs. Dataset 1 was adapted from the publicly available dataset [10][9] - converted to the XES format. However, some of the logs in the original dataset do not follow the specifications described in the referenced paper, i.e., the total of traces. We simulated new event logs using the same process models for these cases. The experiments do not include the "fr" change pattern because it does not represent a structural change. The Petri nets used for the simulator, the simulator source code, and the two datasets are publicly available[10]. For each of the 17 change patterns, dataset 1 included 4 log sizes (2,500, 5,000, 7,500, and 10,000 traces). Each log contains nine drifts injected after 10% of the size (250, 500, 750, and 10,000 traces), changing from the base model to the altered model and vice-versa, simulating nine sudden drifts. All the logs start from the base model and change to the modified.

Dataset 2 has the same change patterns as Dataset 1 and alternates between the base and the modified model. However, it contains 3 log sizes, and the interval between the drifts varies to avoid the bias of a fixed interval between drifts: 3,000 traces with 4 drifts (250; 750; 1,500; and 2,500), 4,500 traces with 7 drifts (250; 750; 1,500; 2,500; 3,250; 3,750; and 4000), and 8,000 traces with 13 drifts (250; 750; 1,500; 2,500; 3,250; 3,750; 4,000; 4,500; 5,250; 6,250; 7,000; 7,500; and 7,750).

For the Adaptive IPDD, we set the ADWIN's $\delta = 0.002$, the default value of ADWIN implementation. We vary the window size (w) for all compared tools[11] between 25 to 300 with a step of 25 traces. In the Apromore ProDrift AWIN, the w value is applied to the initial window size. In VDD, we defined the window step as w / 2 (integer division) and set the clustering option. As VDD does not inform the trace of the drift in the Drift Map, we calculated the trace based on the output information from the console and the w parameter.

[7] https://visual-pro-drift.com.br/
[8] https://github.com/denisesato/InteractiveProcessDriftDetectionFW
[9] https://data.4tu.nl/datasets/0a003285-69d0-4957-9d9f-70f0820066d8
[10] https://github.com/denisesato/SimulateLogsWithDrifts/
[11] We do not include the results using a $w = 25$ because VDD takes too much time to execute (e.g., more than 10 hours for the lo cb10k in an I5-9500 with 8GB RAM).

5 Results and Discussion

5.1 Impact of the w Parameter on the Detection Accuracy

In Fig.3, we plot the average F-score for the 17 change patterns and the four sizes for dataset 1 (2,500; 5,000; 7,500; 10,000 traces). We can observe that the F-score drops after $w = 200$ for the logs containing 2,500 traces (2.5k). This behavior is expected because these event logs contain drifts with an interval of 250 traces, and when setting $w > 200$, the window is too large to detect the change. However, the accuracy of the approach is not affected by smaller w values. For the other log sizes, containing drifts with intervals of 500, 750, and 1,000 traces, the F-score values maintain above 0.8 for all values of w.

Fig.3 also shows the Mean delay, which complements the F-score analysis by indicating how "close" to the actual change point Adaptive IPDD detects the drifts. We can observe in Fig.4 that for the event logs 5k, 7.5k, and 10k, the Mean delay is below 60 traces. In log 2.5k, the Mean delay increases after the w of 200, following the decrease of the F-score, i.e., as the detection accuracy decreases, IPDD also reports drifts more distant than the actual change point.

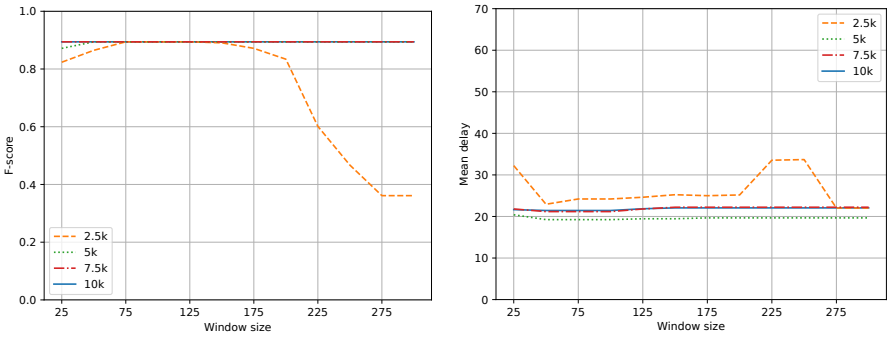


Fig. 3. Dataset 1: Impact of the w parameter on metrics

In Dataset 2, all the logs include an interval between drifts of 250 traces, which explains the drop of the F-score after a w value of 225 - Fig.4. The Mean delay stays stable (close to 40 traces) before w 225, showing that the reported change points are close to the real ones.

5.2 Investigating Detection Accuracy per Change Pattern

We investigate the accuracy per change pattern (Fig.5) to understand better why the average F-score does not reach the value of one even in the "best" considered parameter configuration. Adaptive IPDD never detects the pattern **cd**. This pattern includes a control dependency in the baseline model by synchronizing the **Assess loan risk** with two activities: **Check credit history** and **Appraise property**. The fitness metric does not detect the change to the **cd** model because

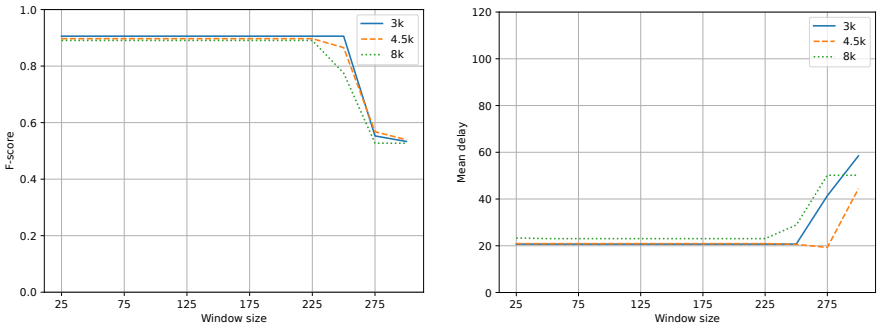
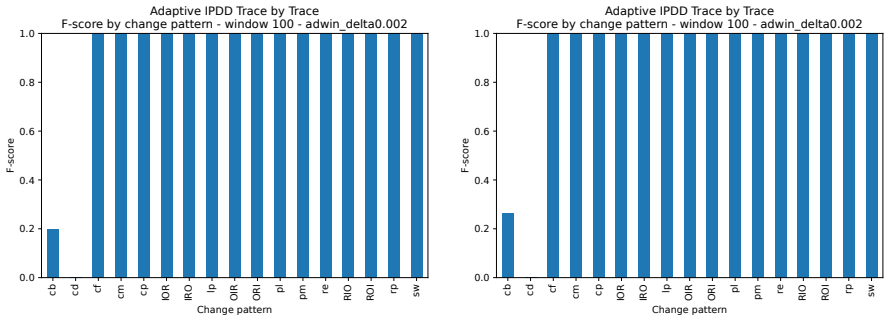


Fig. 4. Dataset 2: Impact of the w parameter on metrics

all possible traces in the **cd** process are also allowed by the base model, i.e., the base model is more generic than **cd**. In this case, we expect the precision metric to detect the change. However, because we evaluate precision using only the last read trace, the decrease in the precision value is not enough to report the change.



(a) Dataset 1 (b) Dataset 2

Fig. 5. Investigating the average F-score per change pattern.

In the case of the **cb** pattern, the changed model includes a silent transition to allow the process instances to skip the sequence of activities **Prepare acceptance pack** and **Check if home insurance quote is requested**. The precision metric evaluates how much behavior the model allows and is not observed in the log. However, when using one trace as the event log, the precision value only changes if the new model is more complex, which is not valid for the **cb** model. A drop in the fitness metric detects the first drift because traces without the two activities decrease the metric's value. However, the precision metric cannot detect when the skip behavior is included. We believe the reason for not detecting drifts in the **cb** and **cd** is related to the precision metric calculation, which considers only the last trace read.

5.3 Comparing Adaptive IPDD with Other Tools

Fig.6 reports the average F-score and Mean delay calculated over the different log sizes. We observe that VDD performed the lowest F-score rates considering both datasets. Adaptive IPDD and Apromore ProDrift achieve higher F-score results; however, the visual analysis does not define if the differences are significant. VDD reports the highest values for the Mean delay, indicating that VDD also reports change points far from the real ones. Also, we observe that the Adaptive IPDD performed a better Mean delay than Apromore ProDrift. The Fixed IPDD reaches F-score 1 but is more sensitive to the w values.

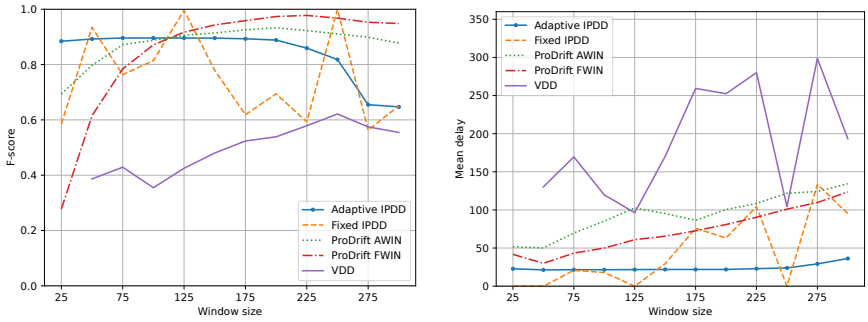
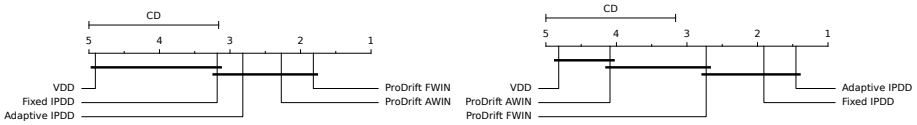


Fig. 6. Metrics for both datasets.

We applied the Autorank library [7] with ($\alpha = 0.05$) to verify whether the observed differences are significant. We excluded $w = 25$ from the analysis because of VDD's results. The Friedmann test rejects the null hypothesis, indicating significant differences between the F-score values and Mean delay. The Nemenyi posthoc test analysis provides the differences between the tools (Fig.7).



(a) F-score (b) Mean delay

Fig. 7. Nemenyi post-hoc analysis.

We can observe that VDD F-score results are significantly lower than Apromore ProDrift and Adaptive IPDD when considering both datasets. However, Adaptive IPDD and Apromore ProDrift produced statistically similar results in terms of accuracy, observed by the F-score analysis. Adaptive IPDD and Apromore ProDrift FWIN presented significantly better results in Mean delay than Apromore ProDrift AWIN and VDD. Also, considering the Mean delay, Adaptive IPDD performance is more stable with the different w configurations (Fig. 6).

6 Evaluation of Adaptive IPDD on Real-Life Event Log

We further evaluated Adaptive IPDD on a public real-life event log representing an Italian company's ticketing management process[12]. The event log contains 4,580 cases and 14 activities from January 2010 to January 2014. The same log was evaluated by [17, 12]. We applied the Adaptive IPDD using a $w = 100$ and $\delta = 0.002$, the best setting obtained in the previous experiments and the same $w = 100$ applied in [17]. Table 1 reports all the detected drifts with the trace index, showing a comparison with VDD and Apromore ProDrift. Based on this comparison, we assume IPDD correctly identified the same two drifts (2 and 4).

Table 1. Comparison between detected drifts in real-life event log

Drift	IPDD	VDD	ProDrift	Considerations
1	1,055	1,000	-	Considered outlier in [17]
2	1,695	1,750	1,716	Detected by all tools
3	2,207	-	-	Only IPDD identify this drift
4	3,903	3,750	3754	Detected by all tools

In a real-life situation, we do not know the ground truth. However, the Adaptive IPDD characterizes each drift by showing the differences between the process models. Based on the provided similarity information, we can verify the characterization of the detected drifts 1 and 3, indicating differences between the derived process models.

Drift 1 - trace 1,055
Nodes similarity: 0.875
Added Nodes: Resolve SW anomaly
Removed Nodes: Schedule intervention
Edges similarity: 0.813
Added Edges: Create SW anomaly → Create SW anomaly, Schedule intervention → Resolve ticket, Take in charge ticket → Schedule intervention, Wait → Resolve ticket
Removed Edges - Assign seriousness → Wait, Closed → Closed

Drift 3 - trace 2,207
Nodes similarity: 0.769
Added Nodes - None
Removed Nodes - Create SW anomaly, Insert ticket, Resolve SW anomaly
Edges similarity: 0.759
Added Edges - Resolve ticket → Wait
Removed Edges - Closed → Closed, Create SW anomaly → Resolve SW anomaly, Insert ticket → Assign seriousness, Resolve SW anomaly → Resolve ticket, Take in charge ticket → Create SW anomaly, Take in charge ticket → Take in charge ticket

[12] Original CSV file was converted to XES using the Disco software

Analyzing drift detection tools in real-life logs raises the challenge of evaluating the results, as we do not know the ground truth. Sometimes, subtle changes may be relevant to the process; other times, they can be considered outlier behavior. Knowing the process and its context, the business analyst may use the information provided to analyze the drifts. We believe that the versions of the process offered by IPDD could help them investigate and validate the results.

7 Conclusion

We presented the Adaptive IPDD, a process drift detection tool. We extensively evaluated IPDD using two synthetic datasets with 17 change patterns and different sizes and intervals between drifts. The datasets and the source code are publicly available. The results are evaluated using F-score and Mean delay metrics, the former measuring the detections' accuracy and the latter assessing if the reported change points are near the real change points.

Adaptive IPDD overcomes VDD and Fixed IPDD considering F-score. We conclude that the accuracy of the Adaptive IPDD approach is affected if the w set is larger than the interval between drifts. Another promising result is that IPDD overcomes VDD's Mean delay in both datasets. Furthermore, Adaptive IPDD provided a similar accuracy to Apromore ProDrift with a better Mean delay than Apromore AWIN. Another important aspect is to reduce the detection sensitivity related to the parameters. Adaptive IPDD is more robust to small w values, decreasing accuracy with w larger than 200 traces. However, the Adaptive IPDD cannot accurately detect patterns that apply a subtle change in the process (**cb** and **cd**). To overcome this limitation, we plan to calculate the precision metric for future work using a sliding window containing the last traces read.

A relevant trait of the experimental protocol is that Adaptive IPDD was validated using synthetic datasets containing sudden drift in the control-flow perspective. Therefore, we plan to investigate Adaptive IPDD further with gradual drifts. The application of Adaptive IPDD in the real-live event log demonstrated that the visualization of the change can enhance the validation and investigation of the drifts in real scenarios. We explored the different process versions to show the detected drifts and their impact on the model. Finally, the two synthetic datasets and the source code of Adaptive IPDD are publicly available.

Acknowledgments. Support and funding by CAPES Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001 (grant numbers 88887.509840/2020-00 and 88887.321450/2019-00).

Disclosure of Interests. The authors have no competing interests to declare relevant to this article's content.

References

1. van der Aalst, W.M.P., Pesic, M., Schonenberg, H.: Declarative workflows: Balancing between flexibility and support. Computer Science - Research and Development **23**, 99–113 (2009)

2. van der Aalst, W.M.P.: Process mining: Data science in action. Springer Berlin Heidelberg (1 2016)
3. Berti, A., van Zelst, S., Schuster, D.: Pm4py: A process mining library for python. Software Impacts **17**, 100556 (2023)
4. Bifet, A., Gavaldà, R.: Learning from time-changing data with adaptive windowing. In: Proceedings of the Seventh SIAM International Conference on Data Mining, April 26-28, 2007, Minneapolis, Minnesota, USA. pp. 443–448. SIAM (2007)
5. Bose, R.P.J.C., van der Aalst, W.M.P., Zliobaite, I., Pechenizkiy, M.: Dealing with concept drifts in process mining. IEEE Transactions on Neural Networks and Learning Systems **25**, 154–171 (1 2014)
6. Gallego-Fontenla, V., Vidal, J.C., Lama, M.: A conformance checking-based approach for sudden drift detection in business processes. IEEE Transactions on Services Computing **16**(01), 13–26 (jan 2023)
7. Herbold, S.: Autorank: A python package for automated ranking of classifiers. Journal of Open Source Software **5**(48), 2173 (2020)
8. Leemans, S.J., Fahland, D., Aalst, W.M.V.D.: Discovering block-structured process models from event logs - a constructive approach. Lecture Notes in Computer Science **7927 LNCS**, 311–329 (2013)
9. Maaradji, A., Dumas, M., Rosa, M.L., Ostovar, A.: Detecting sudden and gradual drifts in business processes from execution traces. IEEE Transactions on Knowledge and Data Engineering **29**, 2140–2154 (2017)
10. Maaradji, A., Dumas, M., La Rosa, M., Ostovar, A.: Fast and accurate business process drift detection. In: Business Process Management. pp. 406–422. Springer International Publishing, Cham (2015)
11. Montiel, J., Halford, M., Mastelini, S.M., Bolmier, G., Sourty, R., Vaysse, R., Zouitine, A., Gomes, H.M., Read, J., Abdessalem, T., Bifet, A.: River: machine learning for streaming data in python. J. Mach. Learn. Res. **22**(1) (jan 2021)
12. Ostovar, A., Leemans, S.J.J., Rosa, M.L.: Robust drift characterization from event streams of business processes. ACM Transactions on Knowledge Discovery from Data **14**, 1–57 (3 2020)
13. Ostovar, A., Maaradji, A., La Rosa, M., ter Hofstede, A.H.M., van Dongen, B.F.V.: Detecting drift from event streams of unpredictable business processes. In: Comyn-Wattiau, I., Tanaka, K., Song, I.Y., Yamamoto, S., Saeki, M. (eds.) Conceptual Modeling. pp. 330–346. Springer International Publishing, Cham (2016)
14. Sato, D.M.V., Barddal, J.P., Scalabrin, E.E.: Interactive process drift detection framework. In: Rutkowski, L., Scherer, R., Korytkowski, M., Pedrycz, W., Tadeusiewicz, R., Zurada, J.M. (eds.) Artificial Intelligence and Soft Computing. pp. 192–204. Springer International Publishing, Cham (2021)
15. Sato, D.M.V., Fontana, R.M., Barddal, J.P., Scalabrin, E.E.: Interactive process drift detection: A framework for visual analysis of process drifts (extended abstract). In: Proceedings of the ICPM Doctoral Consortium and Demo Track 2021 - 10th International Conference on Process Mining (ICPM 2021). pp. 41–42 (2021)
16. Sato, D.M.V., Freitas, S.C.D., Barddal, J.P., Scalabrin, E.E.: A survey on concept drift in process mining. ACM Computing Surveys **54**, 1–38 (2022)
17. Yeshchenko, A., Ciccio, C.D., Mendling, J., Polyvyanyy, A.: Visual drift detection for sequence data analysis of business processes. IEEE Transactions on Visualization and Computer Graphics p. 1 (2021)