Online anomaly detection using statistical leverage for streaming business process events

Jonghyeon Ko and Marco Comuzzi

Department of Industrial Engineering
Ulsan National Institute of Science and Technology (UNIST)
Ulsan, Republic of Korea
`{whd1gus2,mcomuzzi}@unist.ac.kr`

Abstract. While several techniques for detecting trace-level anomalies in event logs in offline settings have appeared recently in the literature, such techniques are currently lacking for online settings. Event log anomaly detection in online settings can be crucial for discovering anomalies in process execution as soon as they occur and, consequently, allowing to promptly take early corrective actions. This paper describes a novel approach to event log anomaly detection on event streams that uses statistical leverage. Leverage has been used extensively in statistics to develop measures to identify outliers and it has been adapted in this paper to the specific scenario of event stream data. The proposed approach has been evaluated on both artificial and real event streams.

1 Introduction

Information logged during the execution of business processes is available in so-called event logs, which contain events belonging to different process instances (or *cases*). Each event is described by multiple attributes, such as a timestamp and a label capturing the activity in the process that was executed.

Event logs are prone to errors, which can stem from a variety of root causes [1,2], such as system malfunctioning or sub-optimal resource behaviour. For instance, sloppy human resources may forget to log the execution of specific activities in a process, or a system reboot may assign a different case id to all the new events recorded after rebooting. Errors in event log hamper the possibility of extracting useful process insights from event log analysis, and should therefore be fixed as early as possible [20].

To this end, the research field of event log anomaly detection (or event log cleaning) has emerged recently, providing methods to detect anomalies at trace level [1,2,9,10,13], i.e., concerning the order and occurrence of activities in a process, and at event level [15,16,18], i.e., concerning the value of attributes of events, using a variety of different approaches. Note that event log anomaly

detection is normally (process) model-agnostic, that is, it does not assume the existence of a process model or clean traces from which a model can be extracted. This aspect separates this research field from traditional process mining research on compliance checking [14].

In the specific case of online settings, i.e., event streams, while research has recently emerged in the field of online compliance checking, only the work by Tavares et al. [19] addresses the issue of anomaly detection. In particular, the authors propose a method to detect point anomalies specified by Principal Component Analysis (PCA). These point anomalies, however, do not normally reflect real-life anomaly patterns, such as inserting, skipping or switching events, commonly considered by event log anomaly detection in offline settings. Therefore, we argue that there is a lot of potential for new research in this area.

More in general, event log anomaly detection in online settings can be crucial for discovering anomalies in process execution as soon as they occur and, consequently, allowing to promptly take early corrective actions. The online settings, however, obviously introduce additional challenges to the design of an event log anomaly detection method. In particular, owing to the finite memory assumption of online settings [4,5,6,19,20], only a limited number of (recent) events are available at any given time to take a decision. This prevents to apply effectively some of the approaches that have been proposed in the literature for event log anomaly detection in offline settings. Probabilistic methods that detect anomalies after having created an intermediate model of frequent process behaviour [1,10,13] are hampered by the fact that only a limited number of events may be available to create such models. Online settings also prevent the application of machine learning reconstructive techniques for anomaly detection, e.g. [16,17]. These, in fact, normally rely on deep learning models, which require a high number of data points (complete process traces in this scenario) to be trained effectively. Also, any update of these models may require a long training time.

In this paper we propose an information-theoretic approach to online event log anomaly detection at trace level. Specifically, we devise an anomaly score based on statistical leverage [11]. The leverage is a relative measure of the information content of observations in a dataset that has been used extensively in statistics to develop observation distance measures and outlier detection techniques. Since leverage captures the information content of one observation in respect of all others in a dataset, the anomaly score proposed in this paper can always be calculated reliably based on the information available at any point in time, resulting in an anomaly detection method that does not require extensive amount of data to be executed effectively.

After having presented the related work (in Section 2), Section 3 presents a trace anomaly score based on the notion of statistical leverage. Then, we discuss how this score can be applied to anomaly detection of streams of events, addressing issues such as the grace period, the finite memory assumption, and the identification of anomaly detection thresholds. The proposed method is evaluated (in Section 4) on both artificial and real event logs injected with trace-level anomalies. Conclusions finally are drawn in Section 5.

2 Related work

While there is only limited literature regarding online event log anomaly detection, a number of recent contributions have focused on online conformance checking. To some extent, conformance checking can be seen as model-aware anomaly detection, since process models, given or extracted from clean traces, can be seen as signatures of positive behaviour to detect anomalies. As referred by [6], there are currently two research lines in online conformance checking: the prefix-alignment approach [20] and the model-based approach [4,5,6].

Conformance checking/alignment of streaming events tends to overestimate the computation of optimal alignments. In order to avoid this issue, [20] provides the first incremental/online conformance checking technique that uses prefix-alignment. Prefix-alignment [20] is characterised by high computational complexity and prevents to define a warming up period. Alternatively, Online Conformance Transition Systema (OCTS) [4,5] can partially check compliance on regions of a process. This technique also suffers from high computational complexity and prevents to consider the warm start scenario. In [6], the first solution to achieve a warm start with streaming events has been proposed by introducing weak order relations, that have reduced computational complexity.

Regarding event log anomaly detection, as mentioned in the Introduction, Tavares et al. [19] have first applied the online clustering algorithm DenStream [7] to detect anomalies on event streams. DenStream clusters cases into two groups, normal and anomalous, using histogram-based frequency of activities contained in each case. Since the histogram-based frequency ignores the sequence of events in traces, DenStream detects point anomalies in event logs defined by Principal Component Analysis (PCA) [21].

3 Research framework

There are two different elements in the proposed framework: the anomaly score and the anomaly detection method. The former (presented in Section 3.1) concerns the definition of a trace anomaly score based on statistical leverage. The latter (Section 3.2) concerns setting a threshold value above which a trace is considered anomalous based on its anomaly score.

3.1 Anomaly score

Statistical *leverage* [11] is a measure indicating how far away each observation is scattered from other observations in a dataset. It has been used as a key support measure for developing different observation distance metrics, such as Cook's distance, the Welsch-Kuh distance, and the Welsch's distance.

Given a matrix X, with $X \in \mathbb{R}^{J \times I}$, of a dataset with J observations and I numerical attributes (or variables), the leverage of the observations in X are the diagonal elements of the projection matrix $H = X(X^T X)^{-1} X^T$. Specifically, the leverage of the j-th observation in X is the diagonal element $h_{j,j} \in H$, which is

Case_ID	Event_ID	Activity_Name	CompleteTimeStamp
	E_{X1}	a_1	2019/10/27 10:25:21
CaseX	E_{X2}	a_2	2019/10/29 12:31:48
	E_{X3}	a_3	2019/10/29 13:01:13
CaseY	E_{Y1}	a_2	2019/10/28 09:41:53
	E_{Y2}	a_4	2019/10/30 14:25:51

Step 1. One-hot encoding

CaseID	EventID	a_1	a_2	a_3	a_4	CompleteTimeStamp
	E_{X1}	1	0	0	0	2019/10/27 10:25:21
CaseX	E_{X2}	0	1	0	0	2019/10/29 12:31:48
	E_{X3}	0	0	1	0	2019/10/29 13:01:13
CaseY	E_{Y1}	0	1	0	0	2019/10/28 09:41:53
	E_{Y2}	0	0	0	1	2019/10/30 14:25:51

Step 2. Concatenate grouped events & 0-padding

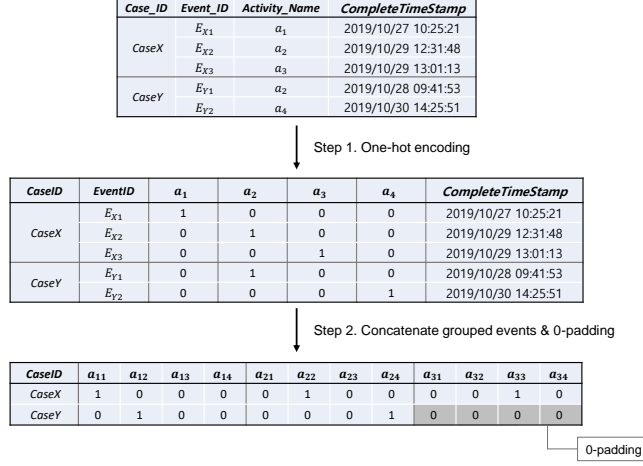| CaseID	a_{11}	a_{12}	a_{13}	a_{14}	a_{21}	a_{22}	a_{23}	a_{24}	a_{31}	a_{32}	a_{33}	a_{34}
CaseX	1	0	0	0	0	1	0	0	0	0	1	0
CaseY	0	1	0	0	0	0	0	1	0	0	0	0

0-padding

Fig. 1: One-hot encoding and 0-padding

comprised by definition between 0 and 1. The higher its leverage, the more likely an observation to be an anomaly.

Our objective is to detect anomalies at the level of occurrence and order of events in traces. Therefore, we can abstract an event log E as a set of J traces $\{\sigma_j\}_{j=1,\ldots,J}$. Each trace is a sequence of events $e_{i,j}$ of variable length N_j, i.e., $\sigma_j = \{e_{1,j}, \ldots, e_{N_j,j}\}$. Events are ordered in a trace by timestamp in ascending order and are defined by the activity that they represent, which is one in a set $A = \{a_1 \ldots, a_K\}$ of K possible activity labels.

In order to define a leverage-based anomaly measure of traces in an event log, two pre-processing steps are necessary. The first one is an integer encoding step. This is necessary because the attributes of a dataset X must be numerical to calculate H, while the activity attribute in event logs is categorical. Second, events in an event log must be aggregated at trace level, such that the resulting matrix X has J rows, i.e., one for each trace. In conclusion, the projection matrix $H(E)$ can be calculated by considering an observation matrix $X(E)$ obtained from E applying the following pre-processing steps.

In the first pre-processing step (see Figure 1), similarly to [16], we apply one-hot encoding, that is, each event $e_{i,j}$ is encoded into a set K dummy attributes $d_{i,j,k}$ such that:

$$d_{i,j,k} = \begin{cases} 1 & \text{if } e_{i,j} = a_k \\ 0 & \text{otherwise} \end{cases}$$

Then, for trace-level aggregation, the one-hot encoded events are horizontally concatenated for each trace. Since traces have different length, for the traces shorter than the longest one(s) in E, i.e., with less events than $N^{max} = \max_{\sigma_j \in E}\{N_j\}$, zero padding is applied. For example, given a case consisting of

4 events and $N^{max} = 5$, the fifth event of this case is zero padded, therefore, $d_{5,j,k} = 0$, $\forall k$. Based on this pre-processing, an event log E is encoded into an observation matrix $X(E)$ with J rows (traces) and $I = N^{max} \times K$ columns (attributes).

Using $X(E)$, we can now define a first leverage-based anomaly score $\hat{l}(\sigma_j)$ by extracting the diagonal elements of $H(E) = X(E) \cdot (X(E)^T \cdot X(E))^{-1} \cdot X(E)^T$:

$$\hat{l}(\sigma_j) = h_{j,j}$$

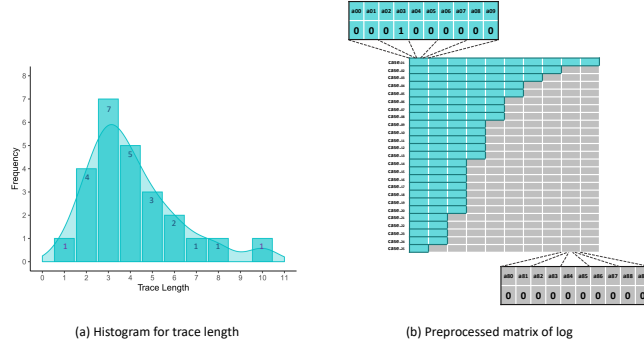(a) Histogram for trace length   (b) Preprocessed matrix of log

Fig. 2: Seesaw effect of zero-padding

This first anomaly score is likely to be biased by the zero-padded attributes in the aggregation pre-processing step. Normally, these zero-padded attributes should be treated as `null` values by any statistical method and therefore not considered in the analysis. However, this is not the case when calculating $\hat{l}(\sigma_j)$. The presence of 0-padded values, as shown in Figure 2, creates a a *seesaw* effect that increases the leverage of longer traces and decrease the one of shorter traces. Shorter traces, in fact, are more likely to be considered similar to each other, and therefore not anomalous, because they are encoded into a higher number of zero-padded values.

In order to counter this issue, we introduce a weighting factor w_j as a function of the trace length to increase/decrease the leverage $\hat{l}(\sigma_j)$ of shorter/longer traces σ_j. This weighting factor is calculated by first normalising the trace length N_j in the range $[0, 1]$. This is done by applying the Z transformation to normalize N_j to the average $mean[N_j]$, followed by the application of a sigmoid function. The sigmoid-based normalisation is generally used to improve the fit accuracy and decrease the computational complexity of the fitting model [12]:

$$sig(Z_j) = \frac{1}{1 + e^{-Z_j}}, \text{ with } Z_j = \frac{\{N_j - mean[N]\}}{stdev[N]} \tag{1}$$

The weighting factor w_j is then defined as:

$$w_j = \left[1 - sig(Z_j)\right]^{c(N^{max})} \tag{2}$$

The power coefficient $c(N^{max})$ is required to adjust the strength of the weighting factor for different event logs. Intuitively, if all traces in a log have similar length, then this adjustment factor should be low, approaching 1; if trace length variance is very high, then the adjustment should be higher.

To define an appropriate value of the power coefficient $c(N^{max})$, a relation between N^{max} and the anomaly detection performance bias should be first found. However, this relation can only be estimated and not optimised because trace length has no upper bound, which would lead to a non-finite state optimisation problem. Therefore, we have estimated the value of $c(N^{max})$ by fitting a non-linear regression using 6 real-life event logs[1] . To model a non-linear regression function, we use the values of $c(N^{max})$ that achieve the highest F1-score in anomaly detection using the 6 different logs in offline settings, i.e., considering all the traces in an event log at the same time in the observation matrix X. Under significance level 0.01, the non-linear equation in Eq. 3 has been fitted with two coefficient parameters a and b as in Table 1.

$$c(N^{max}) = \begin{cases} -2.2822 + (N^{max})^{0.3422} & \text{if } N^{max} > \frac{2.2822}{0.3422} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Table 1: Result of fitted non-linear regression model: $f(x) = a + x^b$

Parameter	Estimate	Standard Error	t value	p-value
a	-2.2822	0.3533	-6.46	0.0030
b	0.3422	0.0191	17.96	0.0001

In the end, using the estimated power coefficient $c(N^{max})$, we define the weighted *leverage*-based anomaly score as:

$$\hat{l}_w(\sigma_l) = w_j \cdot \hat{l}(\sigma_l). \tag{4}$$

3.2 Online anomaly detection

After having defined an anomaly score, the proposed anomaly detection method is complemented by the following four aspects: (i) grace period, (ii) finite memory usage, (iii) update of leverage scores, and (iv) anomaly threshold setting. These are described in detail next.

---

[1] These event logs belong to the ones made available by the Business Process Intelligence Challenge in 2012, 2013 and 2017

Grace period. Similarly to other online anomaly detection methods in the literature [8,19], for practical reasons it makes sense to begin taking decision on trace anomaly only after having received a sufficient number of events. For this purpose, we introduce the parameter Grace Period (GP), which specifies the minimum number of traces and events per trace that must be received before trace anomaly decisions can begin to be taken. In other words, the GP prevents to run the anomaly detection model at early stages, when a sufficient number of events has not been received yet. In this paper, the GP parameter is defined as the number of traces for which at least 2 events have been received. For example, if GP=100, the anomaly detection starts from the first event after having received at least the first 2 events of 100 different traces.

Finite memory usage. Another condition to be satisfied by an anomaly detection method in online settings is the one of finite memory usage. In principle, events may be infinitely received as time goes by. However, handling an infinite number of events would require infinite memory, which is impossible in practical settings [19,20]. Therefore, to calculate leverage using always a finite number of events received from a stream, we introduce the parameter windows size (W), defined as the number of recent cases that are considered to determine anomalies when a new event is received. More formally, at a given time t, let us refer to E_t as the set of events received until t. Now, if the number of (possibly incomplete) traces represented by events in E_t is more than W, then the earliest traces are removed from the set of traces considered to calculate the projection matrix H. Specifically, events of the trace whose first event is the earliest in E_t is first removed and so on until the number of traces represented in E_t is W.

Update of leverage scores. Each time a new event is received, the leverage of the case to which this event belongs is updated (see Figure 3). More in detail, let us assume that an event $e_{i,j}$ is received at time t. Then, after having possibly removed some cases represented by events in E_t to maintain the finite memory usage assumption, the leverage of the remaining traces $\hat{l}_w(\sigma_j)$ is calculated. The result obtained determines, based on the value of the considered anomaly detection threshold, whether the trace σ_j is considered anomalous after the arrival of event $e_{i,j}$, or not. This procedure is replicated each time a new event is received.

Anomaly threshold setting. The objective of anomaly detection is ultimately to determine whether traces are anomalous or not. Therefore, the problem of anomaly detection can be reduced to a binary classification problem. Based on the anomaly score $\hat{l}_w(\sigma_j)$, a decision should be made whether the trace σ_j is anomalous or not. This is normally done by setting the value of an anomaly detection threshold T for $\hat{l}_w(\sigma_j)$, such that a trace is anomalous if $\hat{l}_w(\sigma_j) > T$. In this work, we consider three constant thresholds and one variable threshold. We consider the constant values $T_{c1} = 0.1$, $T_{c2} = 0.15$, and $T_{c3} = 0.2$. These values are based on our experience in experiments in online and offline settings, where anomalous traces tend to have an anomaly score $\hat{l}_w(\sigma_j)$ greater than 0.1. As variable threshold, we consider the value $T_v = mean_{\sigma_j \subseteq E_t}[\hat{l}_w(\sigma_j)] + stdev_{\sigma_j \subseteq E_t}(\hat{l}_w(\sigma_j))$, which calculates
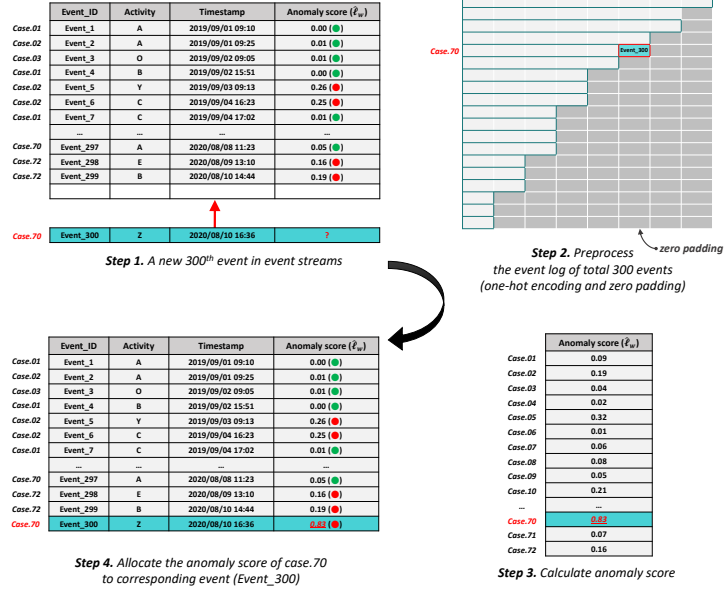
Fig. 3: Example of anomaly detection using a fixed anomaly threshold $T = 0.1$

the threshold based on the mean and standard deviation of the leverage scores of the traces considered in the observation matrix X. A similar principle to set the anomaly detection threshold is used by [16] for the timestamp anomaly detection threshold.

4 Evaluation

This section describes first the datasets that we used for evaluating the proposed online anomaly detection framework. Then, we present the evaluation metrics and experiment settings and, finally, we discuss the performance and computational cost of the proposed framework.

We consider two artificial logs used by [18] and one real-life log publicly available. The artificial logs are generated by simulating 2 process models (Small and Medium in [18]) using the PLG2 tool. Regarding the real log, we consider the *Helpdesk* event log, which contains events logged by a ticketing management system of the help desk of an Italian software company. These logs have been chosen because they have been considered by previous work in anomaly detection and they are also sufficiently small to control the running time of experiments. Descriptive statistics of these logs are reported in Table 2.

Evaluating an unsupervised approach of anomaly detection like the one that we propose requires event logs with labelled traces (normal v. anomalous), which are generally unavailable in practice. Therefore, a common practice in this research field is to inject anomalies using different types of anomaly patterns in

Table 2: Descriptive statistics of event logs (Log statistics are counted after injecting anomalies)

Type	Data	Statistics	Value
Artificial log	Small	-Number of cases	5,000
		-Number of events	44,811
		-Number of activities	20
		-Average # of cases per day	5,000
		-Average # of events per day	44,811
	Medium	-Number of cases	5,000
		-Number of events	29,683
		-Number of activities	32
		-Average # of cases per day	5,000
		-Average # of events per day	29,683
Real log	Helpdesk	-Number of cases	3,804
		-Number of events	13,901
		-Number of activities	9
		-Average # of cases per day	10.94
		-Average # of events per day	18.06

event log and creating labels during the anomaly injection process [1,2,3,16,18]. We consider the 5 anomaly patterns *Skip*, *Insert*, *Early*, *Late*, and *Rework* as defined in [18]: in *Skip*, a sequence of events is skipped in some cases; in *Insert*, one or more events are generated in random positions within existing traces; in *Early/Late*, timestamps of events are manipulated such that a sequence of events is moved earlier/later in a trace; in *Rework*, a sequence of events is repeated after its occurrence. Anomalies are randomly injected in an event log until 10% of the traces in the log have become anomalous. As far as performance measures are concerned, we consider the typical measures for classification problems obtained from the confusion matrix, i.e., precision, recall and F1-score. The datasets used in this paper and the code to reproduce the experiments discussed next are available at `https://github.com/jonghyeonk/OnlineAnomalyDetection`.

We set the GP to 1,000 cases and consider 3 values of window size W, i.e., $W \in [1000, 2000, 3000]$. A larger value of GP and W is likely to lead to better and more stable performance, while also implying a higher computational cost. The experiments are implemented in R on an Intel i7 Linux machine using a single CPU and 5GB memory limit.

Table 3 shows the performance of anomaly detection in event streams for different anomaly detection threshold values and different values of W. Note that the 4 columns T_{c1} to T_v report average performance measures calculated from the start of the streaming (after the GP condition has been reached). It can be noticed that the three constant thresholds show on average better performance than the variable threshold T_v. To better observe a trend of performance improvement as more events are received, the last two columns $T_v^{F:100}$ and $T_v^{L:100}$ show the average performance values calculated on the first 100 events received (after the GP condition has been met) and last 100 events received, respectively. The result shows a clear tendency of increasing performance. The performance is low at the initial stage, and it increases remarkably for the last 100 events received. Particularly in the case of the *Helpdesk* log, the proposed framework could not detect any anomalous cases in the first 100 events, while the performance clearly

Table 3: Performance of online anomaly detection (average from the start of the stream and calculated only on first/last 100 events)

Data	Window size	Time cost (average sec)	Metric	Threshold					
				T_{c1}	T_{c2}	T_{c3}	T_v	$T_v^{F:100}$	$T_v^{L:100}$
Small	1,000	1.09	Precision	0.22	0.22	0.22	0.21	0.06	1.00
			Recall	0.63	0.62	0.59	0.62	1.00	0.63
			F1-score	0.33	0.32	0.32	0.31	0.11	0.77
	2,000	1.07	Precision	0.26	0.25	0.25	0.25	0.06	1.00
			Recall	0.61	0.60	0.57	0.61	1.00	0.63
			F1-score	0.36	0.36	0.35	0.35	0.11	0.77
	3,000	1.23	Precision	0.75	0.79	0.82	0.75	0.20	1.00
			Recall	0.57	0.55	0.53	0.57	0.17	0.63
			F1-score	0.65	0.65	0.64	0.65	0.18	0.77
Medium	1,000	1.42	Precision	0.16	0.17	0.17	0.15	0.50	1.00
			Recall	0.72	0.71	0.71	0.73	0.29	0.50
			F1-score	0.26	0.27	0.27	0.25	0.36	0.67
	2,000	1.46	Precision	0.37	0.45	0.52	0.26	0.50	1.00
			Recall	0.65	0.64	0.63	0.68	0.29	0.50
			F1-score	0.47	0.53	0.57	0.38	0.36	0.67
	3,000	2.06	Precision	0.28	0.30	0.31	0.25	0.50	1.00
			Recall	0.67	0.67	0.66	0.67	0.29	0.50
			F1-score	0.39	0.41	0.42	0.37	0.36	0.67
Helpdesk	1,000	0.34	Precision	0.06	0.06	0.06	0.06	0.00	0.50
			Recall	0.99	0.98	0.96	1.00	0.00	0.96
			F1-score	0.12	0.12	0.12	0.12	0.00	0.66
	2,000	0.37	Precision	0.08	0.08	0.09	0.08	0.00	0.51
			Recall	0.80	0.74	0.68	0.77	0.00	0.96
			F1-score	0.15	0.15	0.15	0.14	0.00	0.67
	3,000	0.39	Precision	0.09	0.11	0.13	0.10	0.00	0.61
			Recall	0.59	0.49	0.39	0.60	0.00	0.93
			F1-score	0.16	0.19	0.20	0.17	0.00	0.74

increases for the last 100 events. It should be noted that, as the number of events received increases, the performance of the proposed framework is likely to converge to the one showed by the average on the last 100 events. Regarding the window size W, the average time cost increases with the value of W. A larger window size also leads to better anomaly detection performance.

As an example, Figure 4 breaks down the performance of the proposed framework along time, counted as the number of events received, in the case of the Small event log with W=3000. It can be noted that the performance oscillates wildly until 20,000 events are received. After that, the performance tends to stabilise and, while the precision remains high, the recall also begins increasing more regularly. After 30,000 events are received, all the performance metrics appear to have become stable.

5   Conclusions

This paper has presented an approach to detect trace level anomalies in business process event streams using an anomaly score based on statistical leverage. A preliminary evaluation on artificial and real event logs also has been presented. The results obtained in this paper are important to determine the future work in this line of research.

First, the performance in the case of the *Helpdesk* log highlights an issue with anomaly threshold setting. The constant values chosen for the experiment
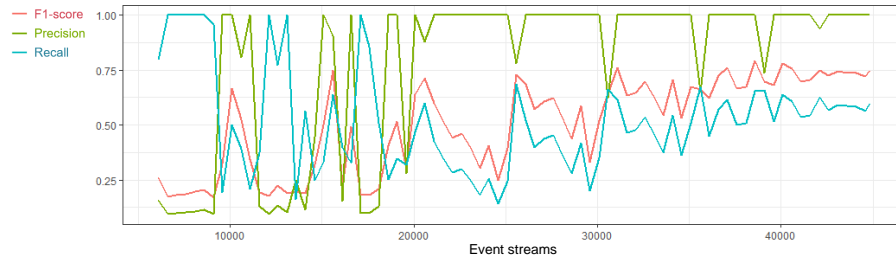
Fig. 4: Performance values as number of events received increases (Small log is applied, using T_v, W=3000)

(between 0.1 and 0.2) appear to be too low for this event log, which results in low precision and F1-score and only high recall. This points to the need for developing an advanced variable threshold that can adapt to the characteristics of different event logs.

Another limitation of the proposed approach, which also impacts the performance, is the fact that it does not distinguish between incomplete and completed traces when calculating the anomaly score. Therefore, many traces may be considered anomalous because they are incomplete, even though they will turn into normal at some point in the future as more events are received. A possible strategy to prevent this is to organise the events received into batches by different prefix length before calculating the anomaly score. This is inspired by [20] that, in the case of online compliance checking, addresses the issue of trace incompleteness using prefix-alignment.

Finally, considering word-embedding instead of one-hot encoding and zero-padding during pre-processing may be likely to reduce the size of the observation matrix X and, therefore, speed up the calculation of anomaly scores.

References

1. F. Bezerra and J. Wainer. Algorithms for anomaly detection of traces in logs of process aware information systems. *Information Systems*, 38(1):33–44, 2013.
2. K. Böhmer and S. Rinderle-Ma. Multi-perspective anomaly detection in business process execution events. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 80–98. Springer, 2016.
3. K. Böhmer and S. Rinderle-Ma. Multi instance anomaly detection in business process executions. In *International Conference on Business Process Management*, pages 77–93. Springer, 2017.
4. A. Burattin. Online conformance checking for petri nets and event streams. In *15th International Conference on Business Process Management (BPM 2017)*, 2017.
5. A. Burattin and J. Carmona. A framework for online conformance checking. In *International Conference on Business Process Management*, pages 165–177. Springer, 2017.

6. A. Burattin, S. J. van Zelst, A. Armas-Cervantes, B. F. van Dongen, and J. Carmona. Online conformance checking using behavioural patterns. In *International Conference on Business Process Management*, pages 250–267. Springer, 2018.

7. 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*, pages 328–339. SIAM, 2006.

8. P. Domingos and G. Hulten. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80, 2000.

9. L. Genga, M. Alizadeh, D. Potena, C. Diamantini, and N. Zannone. Discovering anomalous frequent patterns from partially ordered event logs. *Journal of Intelligent Information Systems*, 51(2):257–300, 2018.

10. L. Ghionna, G. Greco, A. Guzzo, and L. Pontieri. Outlier detection techniques for process mining applications. In *International symposium on methodologies for intelligent systems*, pages 150–159. Springer, 2008.

11. D. C. Hoaglin and R. E. Welsch. The hat matrix in regression and anova. *The American Statistician*, 32(1):17–22, 1978.

12. M. Klimstra and E. P. Zehr. A sigmoid function is the best fit for the ascending limb of the hoffmann reflex recruitment curve. *Experimental brain research*, 186(1):93–105, 2008.

13. S. J. Leemans, D. Fahland, and W. M. van der Aalst. Discovering block-structured process models from event logs containing infrequent behaviour. In *International conference on business process management*, pages 66–78. Springer, 2013.

14. S. J. Leemans, D. Fahland, and W. M. Van der Aalst. Scalable process discovery and conformance checking. *Software & Systems Modeling*, 17(2):599–631, 2018.

15. X. Lu, D. Fahland, F. J. van den Biggelaar, and W. M. van der Aalst. Detecting deviating behaviors without models. In *International Conference on Business Process Management*, pages 126–139. Springer, 2016.

16. H. T. C. Nguyen, S. Lee, J. Kim, J. Ko, and M. Comuzzi. Autoencoders for improving quality of process event logs. *Expert Systems with Applications*, 131:132–147, 2019.

17. T. Nolle, S. Luettgen, A. Seeliger, and M. Mühlhäuser. Analyzing business process anomalies using autoencoders. *Machine Learning*, 107(11):1875–1893, 2018.

18. T. Nolle, S. Luettgen, A. Seeliger, and M. Mühlhäuser. Binet: Multi-perspective business process anomaly classification. *Information Systems*, page 101458, 2019.

19. G. M. Tavares, V. G. T. da Costa, V. E. Martins, P. Ceravolo, and S. Barbon Jr. Leveraging anomaly detection in business process with data stream mining. *iSys-Revista Brasileira de Sistemas de Informação*, 12(1):54–75, 2019.

20. S. J. van Zelst, A. Bolt, M. Hassani, B. F. van Dongen, and W. M. van der Aalst. Online conformance checking: relating event streams to process models using prefix-alignments. *International Journal of Data Science and Analytics*, 8(3):269–284, 2019.

21. S. Wold, K. Esbensen, and P. Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.