2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)

2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS) | 978-1-7281-7649-9/20/$31.00 ©2020 IEEE | DOI: 10.1109/HPCC-SmartCity-DSS50907.2020.00163

Analysing Business Process Anomalies Using Discrete-time Markov chains

Lingkai Yang
School of Computing
Ulster University
Jordanstown, NI, UK
yang-l9@ulster.ac.uk

Sally McClean
School of Computing
Ulster University
Jordanstown, NI, UK
si.mcclean@ulster.ac.uk

Mark Donnelly
School of Computing
Ulster University
Jordanstown, NI, UK
mp.donnelly@ulster.ac.uk

Kashaf Khan
British Telecom
Ipswich, UK
kashaf.khan@bt.com

Kevin Burke
MACSI
University of Limerick
Limerick, Ireland, Kevin.Burke@ul.ie

Abstract*—Within a business context, anomalies can be viewed as indicators for inefficiencies or fraud, which impact upon product quality and customer satisfaction. The development of approaches to monitor, detect and predict anomalous business processes remains an important research topic. In this paper, we propose a method, combining Discrete-time Markov chains (DTMCs) and hitting probabilities (HP), for detecting anomalies occurring in the execution of business processes. Our method extends standard DTMCs to be able to estimate the probability of occurring for a process instance even though it is partially recorded (i.e., the initial executions are missing). The proposed method, denoted as HPDTMC, does not rely on prior knowledge about anomalies and the business process and can be trained on datasets already consisting of anomalies. A Šidák correction is applied to balance the probability of instances of varying length since naturally, process instances with more executions have lower sequence probability and more likely to be detected as anomalies by using DTMCs. We demonstrate the effectiveness of the method by evaluating it on two artificial datasets and one real-life dataset against seven classic anomaly detection methods. In the experiments, our approach reached an F_1 score of 0.904 on average. Moreover, the proposed method outperforms competitors under noisy conditions. The main contribution of this paper is the proposed noise-robust method which is able to detect fully or partially recorded process instances of varying lengths.

***Index Terms*—Process anomaly detection, Discrete-time Markov chains, Hitting probability, Šidák correction**

I. Introduction

Business process anomalies are a natural and inevitable occurrence that often indicate inefficiencies in formal processes, employee skills and customer misunderstandings. Detecting such anomalies can positively impact on the business environment, encouraging process refinement and increasing business efficiency [1]. Typically, a business process represents a series of related executions performed by people or equipment to produce a service or product [2]. Such executions are usually stored as event log files that enable researchers to recreate the process model, explore the patterns of customers and detect process anomalies. In the context of business processes, anomaly detection aims to identify process instances that are significantly different from mainstream behaviour, such as wrongly ordered activities, extremely long execution times and unauthorised devices [3].

Process mining (PM) provides methods to detect anomalies occurring in the execution of a process and one of the most commonly applied techniques is conformance checking [4]. Conformance checking based approaches identify anomalies by comparing process instances to the process model discovered from the event log and instances that do not fit the model are considered as anomalies [5], [6]. In practice, noise filtering methods are usually required to remove infrequent behaviour for avoiding generating complex process models [2].

In this paper, we propose a method combing DTMCs and hitting probabilities for detecting anomalies in business process data. It can be viewed as a conformance checking method, but process instances are fitted to a DTMC model instead of Petri nets or Business Process Model and Notation (BPMN) [1]. By using DTMCs, the ordering of process executions is represented by transition probabilities (TPs) referring to the probability a process instance moves from one activity to another. Therefore, the probability of the occurring (i.e., the sequence probability) of an instance can be estimated by multiplying the initial probability and all subsequent TPs [7]. Process instances with low sequence probability are then considered as anomalies [8]. One of the benefits of our method is it does not require filtering noise or infrequent instances as they naturally relate to low TPs and sequence probabilities.

Moreover, we applied DTMCs to a novel business process scenario where the initial executions of a process instance are not recorded or of our interest. For the former, consider a scenario where customer data in the last six months are stored in a database, and consequently, the initial executions over this period are not recorded. The latter relates to the scenario that we are only interested in customer executions within a particular area. As an example, within a context of online shopping, the areas (AoI) or regions of interest (RoI) might be customer executions after adding items to the shopping cart where anomalies should be inspected [9]. In such scenarios, the observed first activity of a customer is not the initial state of the process leading to difficulties in the estimation of sequence probabilities when using DTMCs. To solve the problem, an extension of DTMCs (namely Hitting Probability-based Discrete-time Markov chain, HPDTMC) is proposed by introducing hitting probabilities to describe the probability of moving from the initial state to the observed (or interesting) first activity. As an example, given a process instance with executions $X_1X_2X_3$, where X_1 is the first observed or interested state. The probability of this instance is therefore $p(X_1) * p(X_2|X_1) * p(X_3|X_2)$ where $p(X_1)$ is given by the hitting probability of getting to X_1.

In summary, the contributions of this paper are as follows.

1. An extension of DTMCs (by introducing hitting probabilities) that is able to estimate the probability of occurring for partially or fully recorded process instances. The method does not rely on prior knowledge about the business process and types of anomalies. As an unsupervised approach, the method can be trained on datasets with noise instances.

2. Novel application scenarios to estimate the sequence probability for partially recorded process instances and then to detect anomalies.

3. Applying the Šidák correction to balance the sequence probability for process instances of varying length. Specifically, to increase the probability of long instances to be detected as normal cases because naturally, they have lower sequence probability as containing more transition probabilities multiplied together and therefore more likely to be detected as anomalies.

The rest of the paper is organised as follows. Section 2 and 3 present related work and the proposed HPDTMC approach, respectively. The findings from a series of experiments along with a discussion of their relevance is presented in Section 4. Finally, conclusions and the scope for future work are provided in Section 5.

II. Related Work

Anomaly detection aims to locate instances in a dataset whose behaviour is unexpected or far away from the mainstream [10]. Techniques to detect business process anomalies can be considered from the perspective of Data Mining (DM), Sequence Mining (SM) and Process Mining (PM) which we now describe.

A. Data Mining

Techniques in the DM community can be classified into two groups: Supervised and Unsupervised. Supervised algorithms address this problem by learning a mapping function between the input vector and the output label. Exemplars include Support Vector Machine (SVM) [11], Decision Trees [10] and Neural Networks [4]. On the other hand, within the anomaly context, unsupervised algorithms examine the structure of the dataset to uncover samples far away from the mainstream, for example, K-means [8] where the samples far away from the clustering centre are considered as anomalies. KNN [8] and Local Outlier Factor (LOF) [12] are two other distance-based approaches. Applying Principal Component Analysis (PCA) [13], sequences can be decomposed into components where those exceeding a threshold are considered as anomalies. One-Class SVM [11] attempts to learn the distribution of normal data so that data outside the learned distribution can be classified as anomalies. As the input size of these models is required to be fixed, extra data pre-processing techniques, such as one-hot encoding [14] or padding [4] should be applied to transform original sequences to the same length.

B. Sequence Mining

Markov- and Window-based methods are commonly used in SM for anomaly detection [8]. In [15], the authors applied a DTMC model to detect intrusions of system calls. In [16], a high-order Markov chain model was used for detecting intrusions in network traffics. An online Markov framework was proposed to detect anomalies in fast streaming data achieving good performance evaluated by the false alarm rate [17]. For a bank financial credit business process, a Hidden Markov Model (HMM) was applied to detect fraud [18]. As Markov-based methods detect anomalies with respect to the context of a set of events, they are able to detect anomalies even for long sequences and simultaneoously locate the position where the anomaly happened [8]. t-STIDE [8] is widely applied as a window-based approach to detect sequence anomalies. The first step of t-STIDE is to separate sequences into windows and to build the profile of normal patterns. During the test stage, a sequence with more windows outside the profile for the defined threshold is detected as an anomalous sequence. Typically, window-based methods are based on the normal profile involving patterns of normal sequences and therefore, the training data is required to be clean (i.e., without noise or anomalies).

C. Process Mining

PM has been proposed to build the bridge between DM and business analysis [1]. Techniques to detect business process anomalies can be divided into two main categories: Model-based and Cluster-based [5]. Typically, the first step of model-based approaches is to discover a process model and thereafter, process instances that do not fit the discovered model are considered as anomalies [6]. On the other hand, Cluster-based techniques classify process instances into groups and instances far away from their clustering centre are considered as outliers [19]. Features can be extracted from the aspects of activities [19], traces [19], durations [20] and repeats [21]. In [4], an autoencoder-based (AE) approach was proposed to detect abnormalities on the activity-level rather than the sequence-level to detect which event influences the anomaly the most. AE is a type of unsupervised neural network where the original input is also used as the target output, and a sequence with reconstruction error higher than the predefined threshold is considered as an anomaly [4].

III. Method

This paper investigates the challenge of identifying particular anomalies where the process instance examined do not necessarily occur at the start of the business process. In practice, it can represent a scenario to detect anomalies in data sampled over a particular time period, such as one week. Consequently, it is assumed that some processes have already commenced within the system (and so the observed first behaviour in the available sample is not the initial state of the business process), whereas other processes may commence for the first time within the available sample (and so the observed first behaviour is the initial state). In this paper, the first scenario is mainly investigated, although the approach also works for the second scenario. We assume that there exist sample customer journey data offering ground truth, initial and terminating states. These data are used as the training set to calculate the initial, hitting and transition probabilities. During the anomaly detection stage, process instances with estimated sequence probability lower than the predefined classification threshold are considered as anomalies.

A. The training stage of HPDTMC

Define a set of process instances, denoted as $X = \{X_1, X_2, ..., X_n\}$. Each instance in X consisting of a series of activities such as $A = A_1A_2...A_k$ where $A \in X$, is considered as a DTMC. The state space and the initial probability of the Markov process is defined as $\boldsymbol{S} = \{s_1, s_2, ..., s_m\}$ and $\boldsymbol{\pi} = \{\pi_{s_1}, \pi_{s_2}, ..., \pi_{s_m}\}$ respectively. The initial probability of state $s_i \in \boldsymbol{S}$, denoted as π_{s_i}, refers to the probability that a DTMC begins from s_i, and can be calculated using

$$\hat{\pi}_{s_i} = \frac{N_{s_i}}{n} \tag{1}$$

where N_{s_i} is the number of instances with s_i as their first state and n is the total number of process instances. The transition probability (TP) describes the probability that a DTMC moves from one state to another in a single step. The TP from state s_i to s_j, $s_i, s_j \in \boldsymbol{S}$ can be calculated using

$$\hat{p}_{s_is_j} = \frac{N_{s_is_j}}{N_{s_i}} \tag{2}$$

$N_{s_is_j}$ is the number of pairs (s_i, s_j) in the whole data set. N_{s_i} is the number of pairs starting with s_i and ending in any one of the states belonging to state space $\boldsymbol{S}$. The hitting probability (HP) [7] relates to the probability the DTMC will ever hit a specified set of states. Specifically, the HP from a particular state to itself equals to one. Otherwise, starting from state s_i, the HP to reach s_j is the sum of the TP from s_i to s_k, $s_k \in \boldsymbol{S}$, multiplied the HP from s_k to s_j. The problem can be solved by using Linear Programming[1].

$$HP(s_i, s_j) = \begin{cases} 1, & s_i = s_j \\ \sum\limits_{k \in S} p_{s_i,s_k} * HP(s_k, s_j), & s_i \neq s_j \end{cases} \tag{3}$$

[1]https://uk.mathworks.com/help/econ/dtmc.hitprob.html

B. The anomaly detection stage of HPDTMC

1) The estimation of sequence probability: Markovian techniques formulate the anomaly detection problem in terms of the probability of the given sequences and therefore, low probability sequences are considered as anomalies [8]. In some scenarios, the initial state of a sequence is not recorded and therefore the observed first element is not the true initial state of the DTMC. Instead, the sequence could start from any one of the possible states in the state space $\boldsymbol{S}$. Consider a full sequence given by $A_1A_2A_3A_4A_5...$, but assume that the customer is in state A_4 when we begin our observation. Then the observed business process is $B_1B_2B_3, ...$ where $B_1 = A_4$ is the observed first state. Because we are missing the initial states $A_1A_2A_3$, traditional DTMCs cannot be used directly to estimate the sequence probability.

As the exact state where a customer started is not recorded, or not of interest, this work assumes it could be any one of the states and all of the potential paths the customer may have experienced are therefore considered. Given a potential initial state (s_i), the hitting probability is applied to describe the possibility that the customer could reach the observed sequence (B_1) from s_i. Thus, the sequence probability is based on the aggregation of all possible initial states and pathways to the first observed state B_1 as shown in equation (4). In other words, each possible path can be calculated by multiplying the initial probability by the hitting probability for the observed first state (B_1), and by all transition probabilities corresponding to the observed transitions ($B_1B_2B_3, ...$). Note that the second line follows from the Markov property meaning the conditional probability distribution of future states of the process depends only upon the present state.

$$\begin{aligned} P(B) &= \sum_{i=1}^{m} \pi_{s_i} \cdot HP(s_i, B_1) \cdot P(B_1, B_2, ..., B_q) \\ &= \sum_{i=1}^{m} \pi_{s_i} \cdot HP(s_i, B_1) \cdot P(B_2|B_1) \cdot ... \cdot P(B_q|B_{q-1}) \\ &= \sum_{i=1}^{m} \pi_{s_i} \cdot HP(s_i, B_1) \cdot \prod_{i=2}^{q} p_{B_{i-1}B_i} \end{aligned} \tag{4}$$

2) Threshold shrinking: The Šidák correction is a method to adjust the significance level in multiple hypothesis testing to control the familywise error rate [22]. In this paper, it is applied to balance the sequence probability for process instances of different lengths. Specifically, the anomaly detection threshold for long sequences is adjusted to be smaller to decrease the probability of being classified as anomalous.

Hypothesis testing is a statistical method used in making decisions involving three main concepts: the null hypothesis (H_0), the alternative hypothesis (H_a) and the significance level (α) [22]. Consider a scenario where we wish to examine the similarity of data in two groups. The null hypothesis states that there is no difference between the two populations. Contrary to H_0, the alternative hypothesis assumes they are different. The significance level refers to the probability that we wrongly reject H_0. However, consider a case where we test 10 hypotheses with α as 0.05. The probability of observing at least one wrong rejection (i.e., rejecting H_0 when it is true) will be $1-(1-0.05)^{10}=0.4$. This kind of hypothesis testing is referred to as multiple hypothesis testing, and the Šidák correction is a commonly used approach to reduce the number of false rejections. It is applied to decrease the significance level for each single hypothesis as shown in equation (5) and therefore to maintain the predefined overall significance level no matter how many hypotheses there are [22].

$$\alpha' = 1-(1-\alpha)^{\frac{1}{q}} \tag{5}$$

Here, α is the overall significance level and α' is the modified significance level in terms of the sequence length q.

In summary, given a process instance, it is detected as an anomaly if its sequence probability (calculated by equation 4) is smaller than the modified anomalous threshold (calculated by equation 5).

C. A case study

As an example, consider a Hospital Billing (HB) event log collected from the local Hospital information systems, involving 18 states and 100000 cases [2]. For simplicity, eleven states were selected to present the main behaviours of the process (as seen in Figure 1 alongside the transition probabilities), covering 97% of the observed data [2].

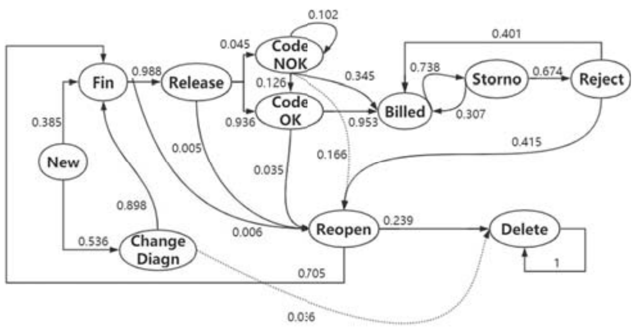


Fig. 1. Control-flow of the Hospital Billing process

The state space is $\boldsymbol{S_{HB}} = \{$‘Billed (BI)’, ‘Change Diagn (CD)’, ‘Code NOK (CN)’, ‘Code OK (CO)’, ‘Delete (D)’, ‘Fin (F)’, ‘New (N)’, ‘Reject (RJ)’, ‘Release (RL)’, ‘Reopen (RO)’, and ‘Storno (S)’$\}$. As patients always start the billing process from state ‘New’, its initial probability π_{New} equals to 1, and others equal to 0. The TPs and HPs are represented in Table I and II, respectively. The HP from ‘Delete’ to ‘Billed’ equals to zero because a patient will never reach ‘Billed’ if they terminate the billing process. The TPs and HPs used in the examples have been highlighted in bold typeset.

Consider three recorded patient journeys represented by sequences as $Seq1 = \{Release, CodeOK, Bill\}$, $Seq2 = \{Release, CodeOK, Reopen, Fin, Release, CodeOK, Bill\}$ and $Seq3 = \{Release, Bill\}$. The first two sequences are viewed as normal as they follow the control-flow of the HB process, as seen in Figure 1. $Seq3$, however, is referred to an anomalous case since patients must check the code before billing. According to equation (4), the sequence probability of $Seq1$ is calculated as,

$$\begin{aligned}
P_{Seq1} &= \sum_{i\in S_{HB}} \pi_i \cdot HP(i,RL)\cdot P(CO|RL)\cdot P(BI|CO)\\
&= \pi_N \cdot HP(N,RL)\cdot P(CO|RL)\cdot P(BI|CO)\\
&= 1.0\cdot 1.0\cdot 0.94\cdot 0.95\\
&= 0.893
\end{aligned}$$

TABLE I
TRANSITION PROBABILITY MATRIX

	BI	CD	CN	CO	D	F	N	RJ	RL	RO	S
BI	0	0	0	0	0	0	0	0	0	0	0.74
CD	0	0	0	0	0	0.90	0	0	0	0	0
CN	0.35	0	0.10	0.13	0	0	0	0	0	0.17	0
CO	**0.95**	0	0	0	0	0	0	0	0	**0.04**	0
D	0	0	0	0	1.00	0	0	0	0	0	0
F	0	0	0	0	0	0	0	0	**0.99**	0	0
N	0	0.54	0	0	0	**0.39**	0	0	0	0	0
RJ	0.40	0	0	0	0	0	0	0	0	0.42	0
RL	**0**	0	0	**0.94**	0	0	0	0	0	0	0
RO	0	0	0	0	0.24	**0.71**	0	0	0	0	0
S	0.31	0	0	0	0	0	0	0.67	0	0	0

TABLE II
HITTING PROBABILITY MATRIX

	BI	CD	CN	CO	D	F	N	RJ	RL	RO	S
BI	1.00	0	0	0.75	1.00	0.75	0	1.00	0.75	1.00	1.00
CD	0.99	1.00	0	1.00	1.00	1.00	0	0.99	1.00	1.00	0.99
CN	0.93	0	1.00	0.80	1.00	0.75	0	0.93	0.75	1.00	0.93
CO	0.99	0	0	1.00	1.00	0.75	0	0.99	0.75	1.00	0.99
D	0	0	0	0	1.00	0	0	0	0	0	0
F	0.99	0	0	1.00	1.00	1.00	0	0.99	1.00	1.00	0.99
N	0.99	0.58	0	1.00	1.00	1.00	**1.00**	0.99	**1.00**	1.00	0.99
RJ	0.87	0	0	0.75	1.00	0.75	0	1.00	0.75	1.00	0.87
RL	0.99	0	0	1.00	1.00	0.75	0	0.99	1.00	1.00	0.99
RO	0.74	0	0	0.75	1.00	0.75	0	0.74	0.75	1.00	0.74
S	0.91	0	0	0.75	1.00	0.75	0	1.00	0.75	1.00	1.00

Similarly, the sequence probability of $Seq2$ is 0.023. Note that the number of activities in $Seq2$ is 7 involving 4 more activities than $Seq1$, while the sequence probability is 38 times smaller showing the huge impact of sequence length on the sequence probability. Consequently, given the classification threshold as 0.05, without the adjustment of threshold, $Seq2$ is wrongly classified as an anomaly. The sequence probability of $Seq3$ is $\pi_N \cdot HP(N,RL)\cdot P(BI|RL) = 1.0\times 1.0\times 0 = 0$, smaller than 0.05 and can be correctly detected as an anomaly.

By applying the Šidák correction, the modified classification threshold (α') for $Seq1$ is $1-(1-0.05)^{\frac{1}{3}} = 0.016$, smaller than the sequence probability and therefore, it is correctly classified as a normal case. As for $Seq2$, the adjusted threshold $\alpha' = 0.007$ smaller than the sequence probability 0.023 and therefore, can be correctly detected as a normal case. Then, consider a patient with fully recorded patient journey $Seq4 = \{New, Fin, Release, CodeOK, Bill\}$. $P_{Seq4} = \pi_N \cdot HP(N,N)\cdot P(F|N)\cdot P(RL|F)\cdot P(CO|RL)\cdot P(BI|CO) = 0.344$ and it can be classified as a normal case correctly. Therefore, the proposed HPDTMC approach can be applied to detect anomalies in partially and fully recorded process instances.

IV. Evaluation

The proposed HPDTMC method was evaluated on two synthetic datasets generated by using PLG [23] and Pykov[2], respectively and also a real-world hospital billing process dataset [2]. The results were compared with seven classic anomaly detection approaches under five types of introduced anomaly including: event switching, skipping, reworking, loop, and inserting, described as follows:

1. Switching [4]. Two events have been executed in the wrong order.
2. Skipping [4]. A necessary event has not been executed.
3. Reworking [4]. An event has been executed multi-times.
4. Loop [21]. A subsequence repeatedly happened.
5. Inserting [24]. An activity is added into the executions.

A. Data set

1) PLG created instances: Process Log Generator (PLG) [23] is software to generate random business processes with different complexity, such as the number of events and the transitions between these events. In this paper, the generated business process involves 19 events and 20 connections without loop structures (presented in Figure 2). Based on the model, process instances referring to normal cases were generated. Thereafter, a set of normal instances were randomly selected and transformed to the five types of anomaly. For Switching, two events were randomly selected and switched while for Skipping, one of the events was randomly deleted simulating the scenario that it was not been executed. For Reworking, an event was randomly selected and executed twice and for Loop, three consecutive events were randomly selected as a group and thereafter they were executed again. Finally, for Inserting, one of the states in state space $\boldsymbol{S}$ was randomly selected and inserted in a random position.

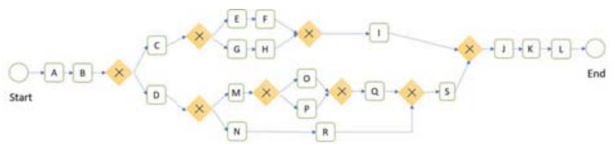


Fig. 2. The BPMN process generated by PLG

To simulate the scenario whereby the initial states of some process instances are missing or not of interest, the second half of the generated instances were used as the period of interest and therefore, anomalies were only created during this period. The fully recorded normal and anomalous process instances were used as the training data of the proposed HPDTMC method. During the testing stage, only the second half of the instances were used. Finally, 8000 normal instances and 750 abnormal instances for each of the five anomaly types were provided as the training and testing data, respectively.

[2]https://github.com/riccardoscalco/Pykov

2) Pykov created instances: Pykov[2] is a Python module employed to generate DTMCs based on the predefined number of states and the initial and transition probabilities. In this paper, we randomly generate a Markov model with 20 states and 50 connections. To simulate the business scenario that a customer can start the business journey from multi-positions, three states are set as the initial states. Consequently, 20000 normal instances with 4067 unique paths of sequence length from 3 to 16 were generated. The same strategy was implemented to generate anomalies, resulting in the generation of 12000 normal sequences and 1500 instances for each of the five anomaly types as the training and testing set, respectively.

3) Hospital Billing event log: The HB [2] event log comprises 100000 process instances recorded over three years with sequence length between 1 and 217 (presented in Figure 1). As no information about sequences type (normal or abnormal) is available, the same strategy used in PLG was applied to generate anomalies. Only those sequences with lengths between 7 and 30 (8435 traces) were used in our simulations and these were of minimum length to extract anomalies while being computationally feasible to accommodate the benchmark classifiers evaluated, where a strategy of padding was used to transform sequences to be the same length as the longest sequence. In total, 4000 normal sequences and 650 abnormal instances for each anomaly types were used.

B. Experimental setup

The proposed method was compared with seven classic anomaly detection methods: AE [4], t-STIDE [8], KNN [8], LOF [12], K-means [8], PCA [13] and One-Class SVM [11]. To evaluate the performance of these methods, k-fold cross-validation [25] was used to execute each algorithm multiple times where k was set to 5. In k-fold cross-validation, the original data set is separated into k equal-sized parts and thereafter, a single part of the data is applied as the validation data for evaluating the model in performance and the remaining k-1 parts of data are used for training. The whole cross validation process is repeated k times with each single part of the data is used once as the validation data. F_1 score was used as the evaluation metric for comparing performance between approaches. To transform sequences to be the same length for PCA, AE, KNN, LOF, K-means and OCSVM, inputs were padded by zeros to be the same size as the longest sequence in the dataset [4]. Thereafter, they were encoded by using one-hot encoding [14]. Consider a business process involving 10 events, and the maximal sequence length is 8. After padding and encoding, each sequence would have a representative size of $10 \times 8 = 80$.

Under the scenario of unsupervised learning, the hyper-parameters in KNN, LOF, OCSVM, etc. are set by using the default or recommended values as shown in Table III. k in KNN and LOF refers to the number of neighbours with n as the number of total instances in the dataset. In K-means, k relates to the number of clusters. α, given as 5%, relates to the proportion of anomalies in the data set, used for deciding the classification boundary. For Autoencoder, one hidden layer was used with nodes as 1/4 of the input size. Batch size, iteration epochs and the learning rate were set as 50, 50 and 0.001 respectively and the network was optimised by Adam [4]. For t-STIDE, a sliding window was first used to build the profile of normal patterns, but patterns with frequency lower than the predefined frequency threshold were excluded from the profile. The window size and the frequency threshold was set as 5 and 0.001, respectively.

TABLE III
HYPER-PARAMETERS AND IMPLEMENTATIONS OF THE SELECTED ALGORITHMS

Algorithm	Platform	Hyper-parameters
KNN	Sklearn [25]	$k \in min(10, 0.4 * n)$, $\alpha = 5\%$
OCSVM	Sklearn	kernel=rbf, degree=3
Kmeans	Sklearn	$k = 2$, $\alpha = 5\%$
LOF	PyOD [26]	$k \in min(10, 0.4 * n)$, $\alpha = 5\%$
PCA	PyOD	$\alpha = 5\%$
AE	Tensorflow [27]	$n_{hidden} = 1/4 \times n_{input}$ batch=50, itera=50, lr=0.001
t-STIDE	Python	$win = 5$, freq=0.001, $\alpha = 5\%$

C. *HPDTMC under different significance levels* (α)

The performance of the proposed approach was evaluated under different α levels. To simulate real scenarios, the training set included 10% anomalous (noise) instances, i.e., 2% for each abovementioned anomaly type. The F_1 scores under different significance levels and datasets are shown as Figure 3. To illustrate the results more clearly, the representative F_1 scores are annotated upon the bar of Pykov dataset.

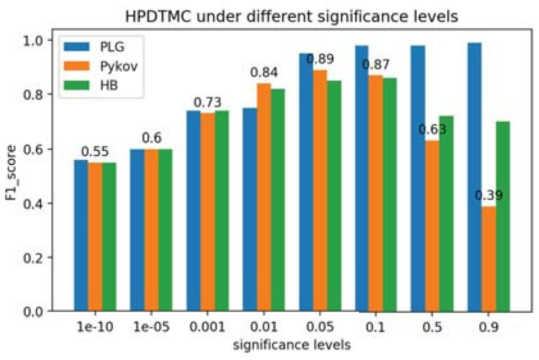


Fig. 3. HPDTMC under different significance levels

According to Figure 3, the performance of HPDTMC varies under different significance levels and the best α is between 0.05 and 0.1. The selection of α depends on the probability of normal and abnormal instances. Take Pykov as an example where the sequence probability of 80% of anomalous instances is smaller than 0.013. Meanwhile, 80% of normal instances with sequence probability larger than 0.068. Therefore, 80% of anomalous and normal cases could be classified correctly as long as the significance level between 0.013 and 0.068. Thus, the significance level was set as 0.05 for subsequent simulations. In practice, under the unsupervised scenarios, we do not have the prior knowledge about the type of anomalies and therefore, α can be specified according to the average value of sequences with lowest probability (for example, the first 1% instances). It is worth highlighting that the actual significance level (α') is smaller than the predefined significance level (α) resulting from the Šidák correction.

D. *Training on normal process instances*

In this part, the training data only contains normal instances for fitting the 8 methods. Table IV describes the F_1 score of all approaches calculated by using the macro average. K-means, OCSVM, t-STIDE and HPDTMC are abbreviated as KM, OCS, tST and HPD respectively. Furthermore, the type of anomaly (Ty), adopts 'SW', 'RE', 'SK', 'LP' and 'IS' to represent switching, reworking, skipping, loop and inserting, respectively. The method performing the best for each scenario is highlighted in bold typeset.

TABLE IV
PERFORMANCE EVALUATION FOR ALL METHODS UNDER DIFFERENT DATASETS AND ANOMALIES

Data	Ty	KNN	LOF	KM	PCA	OCS	AE	tST	HPD
PLG	SW	**1.00**	**1.00**	0.481	0.754	0.787	**1.00**	**1.00**	**1.00**
	LP	**1.00**	**1.00**	0.481	**1.00**	0.820	**1.00**	**1.00**	**1.00**
	RE	**1.00**	**1.00**	0.481	0.894	0.820	**1.00**	**1.00**	0.973
	SK	**1.00**	**1.00**	0.380	0.854	0.735	**1.00**	**1.00**	**1.00**
	IS	**0.93**	**0.93**	0.471	0.753	0.754	**0.93**	**0.93**	0.928
Pykov	SW	0.962	0.954	0.408	0.496	0.480	0.717	**0.994**	0.921
	LP	0.960	0.938	0.515	0.514	0.639	0.735	**0.995**	0.914
	RE	0.965	0.955	0.515	0.460	0.604	0.677	**0.991**	0.801
	SK	0.956	0.947	0.418	0.463	0.438	0.459	**0.988**	0.911
	IS	0.895	0.875	0.582	0.529	0.522	0.745	**0.923**	0.851
HB	SW	0.912	0.837	0.640	0.520	0.508	0.675	**0.946**	0.876
	LP	0.774	0.795	0.611	0.571	0.607	0.554	0.713	**0.803**
	RE	0.875	0.791	0.530	0.555	0.624	0.541	0.879	**0.890**
	SK	0.907	0.837	0.464	0.538	0.426	0.421	**0.942**	0.880
	IS	0.840	0.743	0.510	0.627	0.584	0.531	**0.863**	0.823

The majority of the methods performed well in the PLG dataset, excluding K-means and the best five methods were KNN, LOF, AE, t-STIDE and HPDTMC with F_1 scores above 0.9 for all anomaly types. For Pykov, t-STIDE outperformed all other methods, followed by LOF, KNN and HPDTMC. For HB, the best two methods are t-STIDE and HPDTMC followed by KNN and LOF. Generally speaking, t-STIDE was found to outperform the other methods in almost all of the situations. One of the possible reasons is that t-STIDE extracts normal patterns as a profile and patterns outside the profile are considered as anomalies. During the case that no anomalies are used in the training process, the built profile of normal patterns is 100% correct therefore leading to the outstanding performance. As demonstrated in the subsequent section, t-STIDE performance is greatly affected by the introduction of anomalies during the training stage.

The performance of HPDTMC, LOF and KNN is also satisfactory (with F_1 score above than 0.85 in average) compared to K-means, OCSVM, PCA and AE (with F_1 score of approximately 0.6). One of the reasons for the reduced performance of these latter approaches is their sensitivity to the hyper-parameters which is a common problem in the context of unsupervised learning, especially for AE involving many hyper-parameters [28]. In generally, methods with fewer hyper-parameters are easier to apply because less prior knowledge is needed for parameter specification. In our approach, the only parameter requirement is the overall significance level α, while the other approaches involve at least two hyper-parameters excluding PCA (as presented in Table III).

The experiments were carried out on Windows10 64-bit operating system, 16GB memory, Intel(R) Core(TM) i5-8250U CPU and the average runtime of different methods are demonstrated in Table V. The t-STIDE method requires 365 seconds on average for execution, which is less half consuming time of other methods. On the other hand, the runtime of autoencoder is the largest as it contains greatly more parameters to be optimised. The execution time of our method is close to others excluding t-STIDE.

TABLE V
AVERAGE RUNTIME OF DIFFERENT METHODS (SECONDS)

Method	KNN	LOF	KM	PCA	OCS	AE	tST	HPD
Time(s)	945	985	825	750	795	1020	365	840

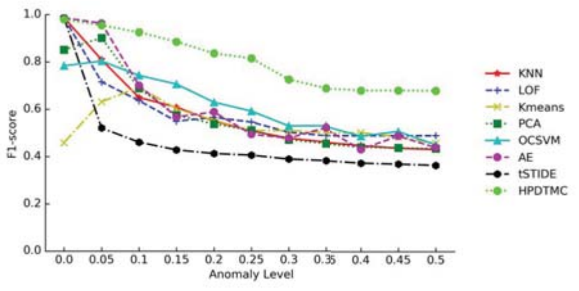
E. Training on data containing noise instances

The performance of the proposed method was evaluated with anomalies introduced during the training process, ranging from 10% to 50% even though noise levels as high as 50% are not common in real-world scenarios [4]. The performance evaluated by F_1 score is shown in Table VI. Furthermore, Figure 4 presents the averaged F_1 score for all methods under different noise levels and datasets. Each value here is an average of F_1 score of the five anomalous types. The proposed HPDTMC method performs satisfactorily even under high noise conditions. While for the other methods, the performance drops dramatically when the noise level is increased. The noise-robust characteristic of HPDTMC approach is due to the robustness of the transition matrix. Specifically, estimated anomalous transitions are still low even with 50% noise during the training process and therefore, the probability of an anomalous sequence remains low; similarly, the probability of a normal sequence remains high.

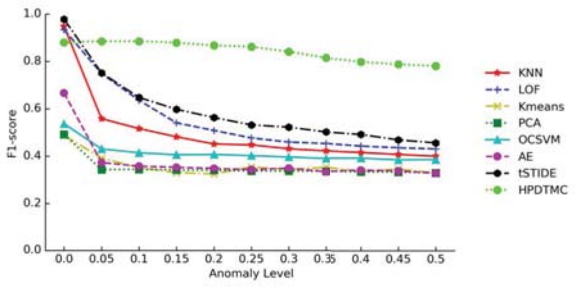
TABLE VI
THE IMPACT OF THE NOISE LEVEL FOR HPDTMC

Data	Ty	0	0.05	0.1	0.2	0.3	0.4	0.5
PLG	SW	1.00	1.00	1.00	0.99	0.93	0.86	0.86
	LP	1.00	0.92	0.77	0.40	0.38	0.31	0.31
	RW	0.97	0.96	0.96	0.94	0.84	0.81	0.80
	SK	1.00	1.00	1.00	1.00	0.64	0.57	0.57
	IS	0.93	0.90	0.90	0.85	0.85	0.85	0.85
Pykov	SW	0.92	0.86	0.86	0.86	0.86	0.85	0.85
	LP	0.91	0.97	0.97	0.92	0.83	0.83	0.82
	RW	0.80	0.79	0.79	0.78	0.75	0.73	0.70
	SK	0.91	0.96	0.96	0.95	0.95	0.77	0.72
	IS	0.85	0.84	0.84	0.83	0.82	0.82	0.81
HB	SW	0.88	0.91	0.92	0.91	0.91	0.89	0.89
	LP	0.80	0.70	0.71	0.71	0.71	0.58	0.58
	RW	0.89	0.90	0.92	0.88	0.88	0.82	0.81
	SK	0.88	0.91	0.91	0.91	0.89	0.88	0.87
	IS	0.82	0.84	0.84	0.83	0.82	0.80	0.79

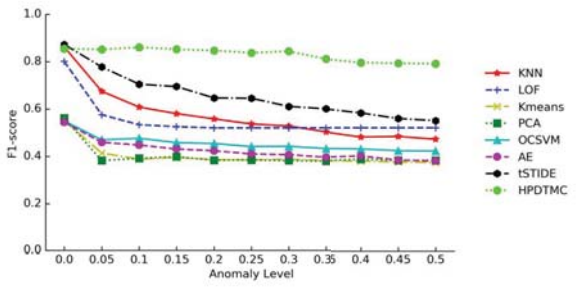
To understand this observation, consider a scenario involving 50 normal sequences and 50 abnormal sequences (overall anomaly level is 50%). As anomalous samples from five types are grouped together, 10% samples from each anomaly type (i.e., 10 sequences) are selected. Suppose the dataset involves 20 different activities (i.e., A, B ,..., T for example) with 50 possible transitions. Moreover, we assume that a customer could only reach B or C if he is located at state A at the current time point with transition probability 0.6, and 0.4 respectively. Therefore, beginning from A, the number of customers who reach B and C is 30 and 20, respectively. For the anomalous cases, suppose customers could arrive at another ten states excluding B and C and therefore, the number of customers reaching one of the anomaly states is 5 on average. Consequently, the transition probability from A to B changes to 0.3, but is still much higher than the transitions referred to as anomalies (0.05 on average).



(a) Averaged F_1 score under dataset PLG



(b) Averaged F_1 score under dataset Pykov



(c) Averaged F_1 score under dataset HB

Fig. 4. Performance of different approaches under different noise-levels

In contrast, the competing approaches, such as KNN and LOF, have a data distribution that is more likely to be changed under high-level noise conditions, leading to poor performance. For t-STIDE, it could extract normal patterns well when the training data is clean. However, under noisy conditions, t-STIDE method overfits to the observed data by treating the anomalies as normal.

Another interesting discovery is the performance of K-means under dataset PLG. The performance of the model is very poor with F_1 score around 0.5 with clean training data. However, after introducing a small number of anomalies, the model performs much better. One of the possible reasons is that the number of clusters is set to two. Therefore, when training with clean data, both clusters compete to identify normal sequences with sequences located at the edge misrepresented as anomalies. Introducing a small number of anomalies into the data promotes a cluster encompassing normal cases as distinct from abnormal sequences that are more likely to be located far away from the clustering centre of normal cases.

V. Conclusion

This paper has proposed a HPDTMC approach to detect anomalies for partially or fully recorded process instances. The approach offers several benefits. The method does not depend on any prior knowledge. It is noise-robust with only one hyper-parameter needing to be specified. The method employs the Šidák correction to modify the threshold for sequences with different lengths reducing the probability that longer sequences are classified as anomalous. The simulation results demonstrated that the proposed HPDTMC approach is a reliable and versatile method for detecting anomalies in the processes environment, in particular, under noisy conditions.

Future work will aim to improve the method via integration of more advanced Markov techniques such as high-order Markov chains, multivariate Markov chains and Hidden Markov models. Furthermore, techniques to detect business process anomalies based on time duration will also be investigated. Also, this approach will be applied to large data representing customer behaviour from the Internet Protocol Television (IPTV) system of British Telecom (BT).

Acknowledgment

This research is supported by BTIIC (the BT Ireland Innovation Centre), funded by BT and Invest Northern Ireland.

References

[1] W. Van Der Aalst, A. Adriansyah, A. K. A. De Medeiros, F. Arcieri, T. Baier, T. Blickle, J. C. Bose, P. Van Den Brand, R. Brandtjen, J. Buijs *et al.*, "Process mining manifesto," in *International Conference on Business Process Management*. Springer, 2011, pp. 169–194.

[2] F. Mannhardt, D. L. Massimiliano, H. A. Reijers, and W. M. P. van der Aalst, "Data-driven process discovery-revealing conditional infrequent behavior from event logs," in *International Conference on Advanced Information Systems Engineering*. Springer, 2017, pp. 545–560.

[3] B. Hompes, J. Buijs, W. Van der Aalst, P. Dixit, and J. Buurman, "Discovering deviating cases and process variants using trace clustering," in *Proceedings of the 27th Benelux Conference on Artificial Intelligence (BNAIC), November*, 2015, pp. 1–9.

[4] T. Nolle, S. Luettgen, A. Seeliger, and M. Mühlhäuser, "Analyzing business process anomalies using autoencoders," *Machine Learning*, vol. 107, no. 11, pp. 1875–1893, 2018.

[5] F. Bezerra and J. Wainer, "Algorithms for anomaly detection of traces in logs of process aware information systems," *Information Systems*, vol. 38, no. 1, pp. 33–44, 2013.

[6] X. Lu, D. Fahland, and W. M. P. van der Aalst, "Conformance checking based on partially ordered event data," in *International conference on business process management*. Springer, 2014, pp. 75–88.

[7] R. Serfozo, *Basics of applied stochastic processes*. Springer Science & Business Media, 2009, pp. 26–30.

[8] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection for discrete sequences: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 5, pp. 823–839, 2010.

[9] R. Es-salhi, I. Daoudi, S. Tallal, H. Medromi *et al.*, "A survey on segmentation techniques of mammogram images," in *International Symposium on Ubiquitous Networking*. Springer, 2016, pp. 545–556.

[10] S. Agrawal and J. Agrawal, "Survey on anomaly detection using data mining techniques," *Procedia Computer Science*, vol. 60, pp. 708–713, 2015.

[11] S. Mahadevan and S. L. Shah, "Fault detection and diagnosis in process data using one-class support vector machines," *Journal of process control*, vol. 19, no. 10, pp. 1627–1639, 2009.

[12] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *ACM sigmod record*, vol. 29, no. 2. ACM, 2000, pp. 93–104.

[13] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing network-wide traffic anomalies," in *ACM SIGCOMM computer communication review*, vol. 34, no. 4. ACM, 2004, pp. 219–230.

[14] K. K. Yang, Z. Wu, C. N. Bedbrook, and F. H. Arnold, "Learned protein embeddings for machine learning," *Bioinformatics*, vol. 34, no. 15, pp. 2642–2648, 2018.

[15] T. Xinguang, D. Miyi, S. Chunlai, and L. Wenfa, "Intrusion detection based on system calls and homogeneous markov chains," *Journal of systems engineering and electronics*, vol. 19, no. 3, pp. 598–605, 2008.

[16] W. Sha, Y. Zhu, T. Huang, M. Qiu, Y. Zhu, and Q. Zhang, "A multi-order markov chain based scheme for anomaly detection," in *2013 IEEE 37th Annual Computer Software and Applications Conference Workshops*. IEEE, 2013, pp. 83–88.

[17] H. Ozkan, F. Ozkan, and S. S. Kozat, "Online anomaly detection under markov statistics with controllable type-i error," *IEEE Transactions on Signal Processing*, vol. 64, no. 6, pp. 1435–1445, 2015.

[18] D. Rahmawati, R. Sarno, C. Fatichah, and D. Sunaryono, "Fraud detection on event log of bank financial credit business process using hidden markov model algorithm," in *2017 3rd International Conference on Science in Information Technology (ICSITech)*. IEEE, 2017, pp. 35–40.

[19] M. Goldstein and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data," *PloS one*, vol. 11, no. 4, p. e0152173, 2016.

[20] M. Song, C. W. Günther, and W. M. P. Van der Aalst, "Trace clustering in process mining," in *International Conference on Business Process Management*. Springer, 2008, pp. 109–120.

[21] R. P. J. C. Bose and W. M. P. van der Aalst, "Trace clustering based on conserved patterns: Towards achieving better process models," in *International Conference on Business Process Management*. Springer, 2009, pp. 170–181.

[22] H. Abdi, "Bonferroni and šidák corrections for multiple comparisons," *Encyclopedia of measurement and statistics*, vol. 3, pp. 103–107, 2007.

[23] A. Burattin, "Plg2: Multiperspective process randomization with online and offline simulations." in *Online Proceedings of the BPM Demo Track*, 2016, pp. 1–6.

[24] 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"*. Springer, 2016, pp. 80–98.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[26] Y. Zhao, Z. Nasrullah, and Z. Li, "Pyod: A python toolbox for scalable outlier detection," *Journal of Machine Learning Research*, vol. 20, no. 96, pp. 1–7, 2019. [Online]. Available: http://jmlr.org/papers/v20/19-011.html

[27] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[28] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in neural information processing systems*, 2011, pp. 2546–2554.