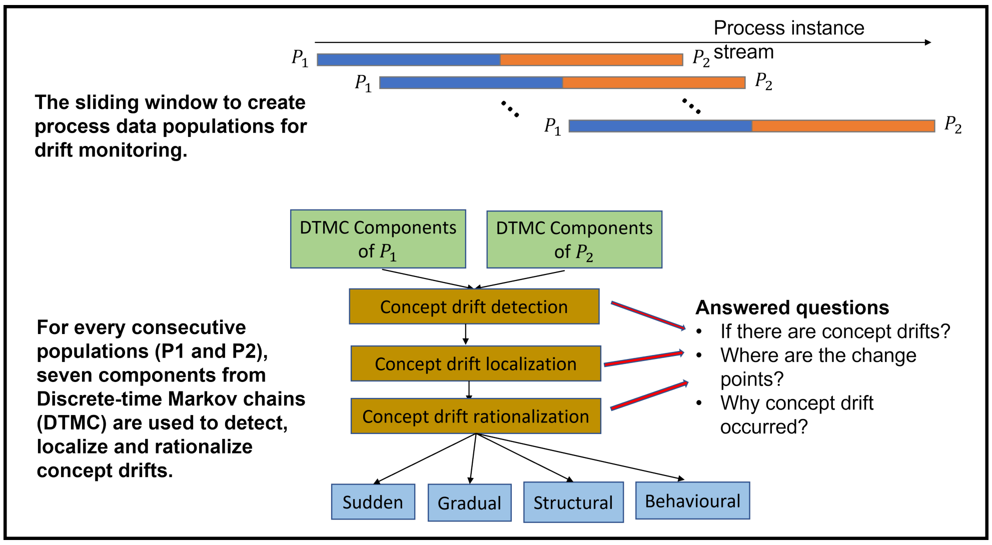
A multi-components approach to monitoring process structure and customer behaviour concept drift

Graphical Abstract

A Multi-Components Approach to Monitoring Process Structure and Customer Behaviour Concept Drift

Lingkai Yang, Sally McClean, Mark Donnelly, Kevin Burke, Kashaf Khan


Process instance stream
P_1
P_2
The sliding window to create process data populations for drift monitoring.
DTMC Components of P_1
DTMC Components of P_2
Concept drift detection
Concept drift localization
Concept drift rationalization
Answered questions
If there are concept drifts?
Where are the change points?
Why concept drift occurred?
For every consecutive populations (P1 and P2), seven components from Discrete-time Markov chains (DTMC) are used to detect, localize and rationalize concept drifts.
Sudden
Gradual
Structural
Behavioural

Highlights

A Multi-Components Approach to Monitoring Process Structure and Customer Behaviour Concept Drift

Lingkai Yang, Sally McClean, Mark Donnelly, Kevin Burke, Kashaf Khan

- A method for detecting sudden and gradual drifts in process event data.
- The method can detect structural and behavioural process drifts.
- A sliding window framework associated with the proposed drift detector.
- The method can detect, localize and rationalize process drifts.

A Multi-Components Approach to Monitoring Process Structure and Customer Behaviour Concept Drift

Lingkai Yang[a,*], Sally McClean[a], Mark Donnelly[a], Kevin Burke[b], Kashaf Khan[c]

[a]*School of Computing, Ulster University, Jordanstown, BT37 0QB, Northern Ireland, United Kingdom*
[b]*Mathematics Applications Consortium for Science and Industry, University of Limerick, V94 T9PX, Limerick, Ireland*
[c]*British Telecom, Ipswich, IP5 3RE, United Kingdom*

*Corresponding author

Email addresses: `yang-l9@ulster.ac.uk` (Lingkai Yang), `si.mcclean@ulster.ac.uk` (Sally McClean), `mp.donnelly@ulster.ac.uk` (Mark Donnelly), `Kevin.Burke@ul.ie` (Kevin Burke), `kashaf.khan@bt.com` (Kashaf Khan)

Abstract

Concept drifts within business processes are viewed as variations in the business circumstances, such as structural and behavioural changes in the control-flow, which necessitate process refinement and model updating. Existing approaches, such as relation-based precedence rules, tuned to detect drifts in the process structure are often not well suited to detecting changes in customer behaviour. This paper proposes a concept drift detector employing multi-components originating from Discrete-time Markov chains to detect, localize and reason about concept drifts in both process structure and customer behaviour of the control-flow. The approach was compared with three commonly used methods using 52 artificial event logs representing various types of drift (sudden and gradual, structural and behavioural). Experimental results demonstrated desirable performance with average F1 scores of 0.871 and 0.893 under structural and behavioural drifts, respectively. The approach was also employed in a real-life hospital billing dataset. The main contribution of this paper is a concept drift detector that is able to detect and explain root causes of control-flow changes whether such variations occurred suddenly or gradually.

Keywords: Business process, Concept drift, Behavioural drift,
Discrete-time Markov chains, Sliding window.

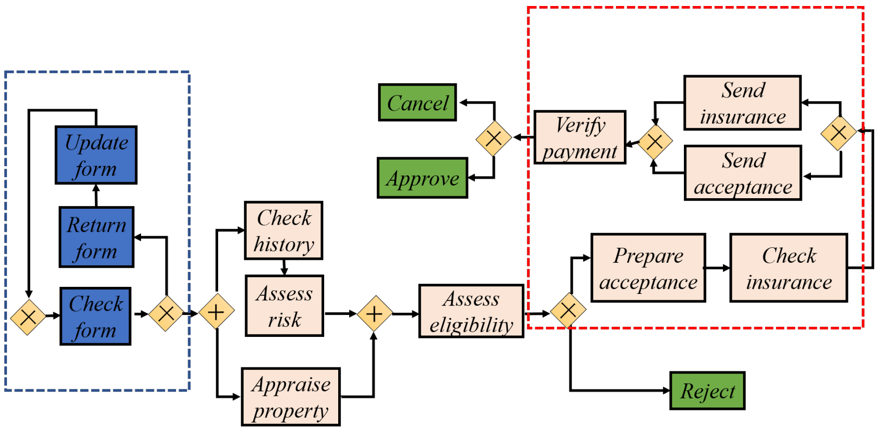
1. Introduction

A process instance, within the context of a business process, refers to a series of process tasks executed in order to accomplish a predetermined objective (Bose et al., 2013). Business environments may evolve continuously or under some scenarios, suddenly due to variations in, for example, policies, customer behaviours, supply chains, data or available resources. This presents ongoing challenges for organizations as they need to quickly respond and adapt (Maaradji et al., 2017). Within the data mining community, such changes are referred to as concept drifts (Gama et al., 2014) and have been classified into four main categories: sudden, gradual, recurring and incremental (Gama et al., 2014; Maisenbacher & Weidlich, 2017). In general, a concept drift is sudden if the business environment changes instantly without alteration (Gama et al., 2014), while for gradual drift, the current business process is replaced with a new process model, but both processes coexist for a period of time with the current process discontinuing gradually (Bose et al., 2013). The recurring concept relates to scenarios where previously seen concepts reoccur after some time (Maisenbacher & Weidlich, 2017) and in this paper, we consider it as a special case of sudden changes. Incremental drift refers to a sequence of changes in the environment (Gama et al., 2014) that can be viewed as a series of gradual drifts.

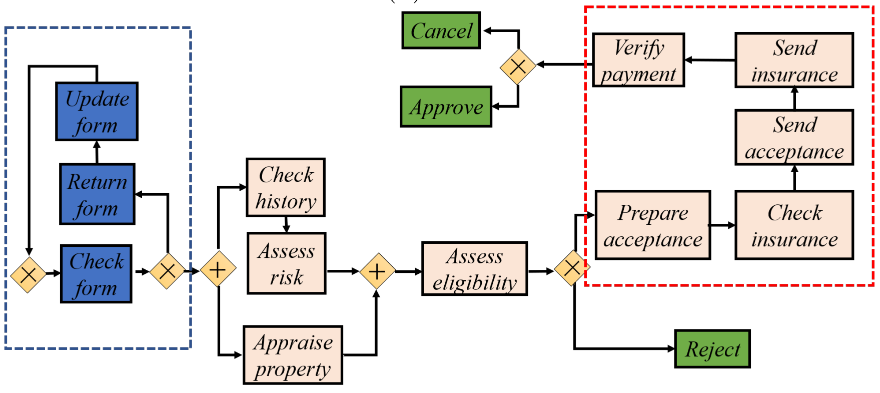
These drifts may impact upon control-flows; structural or behavioural changes in a process model (Bose et al., 2013). By structural changes, we mean variations in the process model, such as the start or end of the process, or the sequencing order of executions. For example, a temperature check procedure is introduced to a supermarket to keep safe during the COVID-19 epidemic (Velavan & Meyer, 2020). By behavioural changes, we say that the structure of a process remains the same, but the patterns or pathways of customers traversing the process are significantly different (Martjushev et al., 2015). For example, customers may begin bringing their own bags instead of buying a plastic bag when they go shopping or, due to a sudden factor, for example, the COVID-19, customers become more likely to obtain fresh products via delivery rather than travel to the store. Excluding control-flow, concept drifts can also occur from a data or performance perspective. The former refers to changes in the production and consumption of data (Bose et al., 2013) while the latter focuses on the validation of the execution of activities, such as the waiting time in-between activities (Rojas et al., 2016). Detecting such changes can provide insight into the evolution of a business environment, encouraging process refinement and model updating.

Martjushev et al. (2015) identified three main objectives when dealing with business process drifts: change point detection (whether concept drifts happened?), change localization (where are the regions of changes?) and change process discovery (why the changes occurred?). Concept drift detection approaches can be broadly classified into online and offline depending on whether or not the occurrence of changes needs to be discovered in real-time (Martjushev et al., 2015). Maaradji et al. (2017) generated a benchmark of 18 types of control-flow structural sudden changes including loops, parallel and alternative branches by altering the structure of a loan application process. The original process model is denoted as 'base', as illustrated in Fig. 1 (a) and one of the altered models referred to as 'cf', is illustrated in Fig. 1 (b). The three activities in blue boxes refer to a loop structure. Process instances start from 'Check form' and terminate in one of the green activities.

Suppose the base model was replaced by the altered model at a particular time point, but such a change is unknown to us. Therefore, the question we consider is if we can detect and locate this change and also, explain the reasons (i.e., the variations in the red boxes between Fig. 1 a and b) based on a collection of process instances?



(a) base model



(b) altered model (‘cf’)

Figure 1: The base model and one of its alternative versions

Stochastic business processes can be modelled using Discrete-time Markov Chains (DTMCs). In this paper, we use seven components of DTMCs to build an offline concept drift detector, referred to as DTMC-CDD (DTMC Components-based Concept Drift Detector), focused on detecting both structural and behavioural changes in control-flow. The seven components comprise the state space, initial and transition probabilities, communication, recurrent and periodic classes, and absorbing states. A sliding window-based framework is applied alongside DTMC-CDD to localize and rationalize identified change points.

In summary, the contributions of this paper are as follows. 1) The proposed DTMC-CDD method, which combines seven components originating in DTMCs as a concept drift detector for process mining. As an offline approach, it is available for dealing with sudden and gradual concept drift. Also, the approach is capable of detecting structural changes in control-flow such as loop structure, start and end of the process, as well as behavioural changes. 2) The applied sliding window framework associated with the proposed DTMC-CDD approach to partition process instance stream into consecutive populations for change point detection and localization. 3) The 34 created logs by modifying the loan application (Maaradji et al., 2017), including 18 logs referring to gradual changes in the control-flow structure and 8 logs each representing behavioural drifts occurring suddenly or gradually. Those datasets have been uploaded to Github[1].

The rest of the paper is organized as follows. Section 2 and 3 present related work and the background knowledge of business processes and DTMCs. We introduce the proposed DTMC-CDD method and the sliding window framework in Section 4. The findings from a series of experiments on artificial and real-life event logs are presented in Section 5 and 6, respectively. Section 7 discusses the strengths and weaknesses of the proposed approach. Finally, conclusions and the scope for future work are provided in Section 8.

2. Related Work

Process mining (PM), identified as a bridge between data mining and business process analysis, includes various techniques to monitor business process executions, discover business bottlenecks and detect concept drift (Bose et al., 2013). In this section, we compare our proposed DTMC-CDD methodology to state-of-the-art process drift detection approaches (as demonstrated in Table 1) from four perspectives: regarded perspectives (i.e., changes in process structure, behaviour, data or performance), the application scenarios (i.e., online or offline), the patterns of change (i.e., sudden or gradual) and the topics of study (i.e., process drift detection, localization or rationalization).

[1]https://github.com/LingkaiYang/business-concept-drift

Table 1: Method comparison in regarded perspectives, application scenarios, patterns of change and topics

References	Regarded Perspectives				Scenarios		Patterns of Change		Topics		
	Str	Beh	Dat	Per	On	Off	Sud	Gra	Det	Loc	Rat
Bose & van der Aalst, 2009	✓					✓	✓		✓		
Bose et al., 2013	✓					✓	✓	✓	✓	✓	
Hompes et al., 2015	✓	✓	✓			✓	✓		✓		✓
Martjushev et al., 2015	✓					✓	✓	✓	✓	✓	✓
Seeliger et al., 2017	✓					✓	✓		✓	✓	
Maisenbacher & Weidlich, 2017	✓		✓		✓		✓	✓	✓		
Ostovar et al., 2017	✓				✓		✓		✓		✓
Zheng et al., 2017	✓					✓	✓		✓	✓	
Maaradji et al., 2017	✓	✓			✓		✓	✓	✓		
Yeshchenko et al., 2019	✓					✓	✓	✓	✓		✓
Tavares et al., 2019	✓	✓		✓	✓		✓	✓	✓		
Ceravolo et al., 2020	✓		✓	✓	✓		✓	✓	✓		
Ostovar et al., 2020	✓				✓	✓	✓		✓		✓
Adams et al., 2021	✓			✓		✓	✓		✓		✓
DTMC-CDD (this paper)	✓	✓				✓	✓	✓	✓	✓	✓

Str=Structure, Beh=Behaviour, Dat=Data, Per=Performance
On=Online, Off=Offline
Sud=Sudden, Gra=Gradual
Del=Detection, Loc=Localization, Rat=Rationalization

In general, the majority of methods concentrate on detecting process structural drifts, such as the insertion or deletion of a process, while other perspectives are not fully studied. Hompes et al. (2015), Maaradji et al. (2017) and Tavares et al. (2019) discussed process behavioural changes, but they mainly concentrate on variations in the trace-level, i.e., changes in the proportions of different activity sequences. Moreover, only the method proposed by Hompes et al. (2015) analyzes the root causes of behavioural drift. By using DTMC transition probabilities, our proposed method can detect behavioural drift at an event-level to discover which specific process or subprocesses lead to the changes. As a result, the method can investigate deeper the root causes of the drift. In addition, Martjushev et al.'s (2015) and our method are the only two techniques to detect, localize and rationalize concept drift, whether it occurs suddenly or gradually.

Offline learning approaches collect all process instances for model building, which is beneficial for robust data characterizing and understanding. Sliding windows are commonly applied in various concept drift detection approaches to generate a set of populations (Bose et al., 2013; Martjushev et al., 2015) where changes are examined between consecutive pairs. Bose et al. (2013) proposed four features based on the follows/precedes relationship (i.e., for any pair of activities 'a' and 'b', activity 'a' always, sometimes or never follows activity 'b'). Thereafter, Hotelling T^2, KS and MW tests were used to expose changes. In Seeliger et al. (2017), process instances were transformed to graph metrics, such as the number of edges and nodes, and a G-test was used as the statistical test method. Clustering methods such as Agglomerative Hierarchical Clustering (Bose & Van Der Aalst, 2009), Markov Clustering (Hompes et al., 2015) and Density-based Spatial Clustering of Applications with Noise (DBSCAN) (Zheng et al., 2017) are also commonly applied to detect process drift. Yeshchenko et al. (2019) introduced a visual technique, called Visual Drift Detection (VDD), to handle the issues of drift categorization, drilling-down, and quantification. Adams et al. (2021) addressed the importance of understanding the root causes of process drift, especially for decision making.

Online learning approaches, on the other hand, ingest one observation data at a time, with the capability to update continuously and identify drift in real-time. Ostovar et al. (2017) proposed an online technique for the early detection of business process changes. Thereafter, they presented a noise-tolerant automated method, which is capable to expose concept drift in both online and offline scenarios. In addition to detecting concept drift in the control-flow of a business process, Maisenbacher & Weidlich (2017) presented an online approach to investigate whether the detected changes actually influence the prediction of properties of process instances. Maaradji et al. (2017) reorganized process instances as RUNs, characterizing the pathways of customers and then applied the Chi-square test to uncover variations. Tavares et al. (2019) proposed an approach to simultaneously handle multiple process mining tasks, such as process discovery, conformance checking, and concept drift detection. Ceravolo et al. (2020) discussed the state of the art in online process mining, highlighting the evaluation goals and event logs for concept drift detection. Ostovar et al. (2020) proposed a robust and automated method that works both online and offline, with the goal of characterizing process drifts in streams of business process events.

A Markov process is a stochastic process that is memoryless to historical data, thus the future of the process is dependent only upon its present value (Meyn & Tweedie, 2012). Regarding drift detection, Roveri (2019) proposed three strategies by using transition probabilities to investigate changes, depending on whether or not prior knowledge is required. Alippi et al. (2012) applied hidden Markov models (HMMs) to capture the distribution of two-time series, respectively where a significant difference evaluated by using the log-likelihood, refers to concept drift. Borges & Levene (2008) applied a variable-length Markov chain (MC) to detect concept drifts in a context of web usage mining where a concept drift is detected if the performance in two trained MC models (depending on the whole history data and currently recorded data, respectively) is significantly different. In our previous work, DTMCs and hitting probabilities are applied to detect business process anomalies where changes in the anomalous patterns can be indicative of concept drift (Yang et al., 2020).

3. Preliminaries

This section introduces the formal preliminaries used throughout the paper. Section 3.1 describes the event log. Sections 3.2 and 3.3 detail the Markov components used in this paper from perspectives of probability distribution (initial and transition probabilities) and state classification (communication, recurrent, periodic and absorbing states).

3.1. Business process event log

Definition 3.1. (Event log) An event log L is a set of recorded process instances that correspond to process executions (Yeshchenko et al., 2019). we present the set of executions of a process as a finite non-empty alphabet $A = \{a, b, c, d, \cdots\}$. An event is a specific process execution, commonly with completion timestamp, resource and other attributes. A process instance x is a finite sequence of events, sorted by their timestamps. For the sake of simplicity, we consider x as a sequence of executions, for example, $x = abca$.

3.2. DTMC and probability distribution

Definition 3.2. (DTMC) A stochastic process $(X_0, X_1, X_2, ..., X_n,\ n \geq 0)$ is a DTMC with a state space $(S = \{s_1, s_2, ..., s_k\})$, an initial distribution $(\pi = \{\pi_1, \pi_2, ..., \pi_k\})$ and a transition probability matrix $P = (p_{ij}, i, j \in S)$ if it holds the Markov property,

$$P(X_{n+1} = j | X_0, ..., X_n = i) = P(X_{n+1} = j | X_n = i). \tag{1}$$

That is, the conditional probability distribution of future states only depends on the current state. Although in some scenarios this assumption is not realistic, it is generally accepted as an approximation in the context of business processes . Proofs for the following definitions and theorems can be found in (Serfozo, 2009; Sartea et al., 2019).

Definition 3.3. (State space) The state space is a countable set of unique states, denoted as $S = \{s_1, s_2, \cdots, s_k\}$. Therefore, changes in S usually relate to operations, such as inserting or deleting a process execution.

Definition 3.4. (Initial probability) The initial probability of state s_i, denoted as π_{s_i}, refers to the probability that a DTMC begins from s_i, and is estimated, using maximum likelihood estimation, as:

$$\hat{\pi}_{s_i} = \frac{N_{s_i}}{N} \tag{2}$$

where N_{s_i} is the number of sequences with s_i as their first state and N is the total number of sequences. Thus, initial probabilities can expose changes at the beginning of process instances.

Definition 3.5. (Transition probability) The transition probability $p_{s_i s_j}$ describes the probability that a DTMC moves from state s_i to s_j in a single step which is given as:

$$\hat{p}_{s_i s_j} = \frac{N_{s_i s_j}}{\sum_{v \in S} N_{s_i s_v}}. \tag{3}$$

$N_{s_i s_j}$ is the number of pairs (s_i, s_j) in the event log. $\sum_{v \in S} N_{s_i s_v}$ is the number of transitions that start with s_i and end in any one of the states belonging to the state space S. The transition probabilities can be listed in a $k \times k$ transition probability matrix, denoted as P where $P(i,j) = p_{s_i s_j}, i = 1, ..., k, j = 1, ..., k$. Variations in the transition matrix is indicative of possible concept drift in customer behaviour.

Definition 3.6. (n-step transition probability) The n-step transition probability $p_{s_i s_j}(n)$ is the probability of making a transition from state s_i to s_j over n steps, i.e., $p_{s_i s_j}(n) = P(X_{r+n} = s_j | X_r = s_i)$.

Theorem 3.1. *(Chapman-Kolmogorov equations)*

$$p_{s_i s_j}(m+n) = \sum_r p_{s_i s_r}(m) p_{s_r s_j}(n). \tag{4}$$

Let $P_n = \{p_{s_i s_j}(n)\}$ be the n step transition matrix, and $P_1 = P$. According to **Theorem** 3.1, $P_{m+n} = P_m P_n$, and so $P_n = P^n$, the n-th power of P.

3.3. Classification of DTMC states

Definition 3.7. (Reachability) State s_j is said to be reachable from s_i ($s_i \to s_j$) if starting from s_i, it is possible to reach s_j for some transition steps n, i.e., $p_{s_i s_j}(n) > 0$. This means $s_i \to s_j$ if $\sum_{r=0}^{\infty} p_{s_i s_j}(r) > 0$, indicating that we need to compute the n-step transition probability for every value of r up to infinity. In practice, only those values of r between 0 and $k-1$ is of our interest as a path of $k-1$ transitions involves all the states in the process. Thus, $s_i \to s_j$ if,

$$\sum_{r=0}^{k-1} p_{s_i s_j}(r) > 0. \tag{5}$$

Definition 3.8. (Communication class) s_i and s_j are in the same communication class if $s_i \to s_j$ and $s_j \to s_i$. Thus, in the same communication class, every state is reachable from every other state and in the business process context, refers to states with a loop structure. To partition states into communication classes, we first establish a $k \times k$ matrix Q,

$$Q(s_i, s_j) = \begin{cases} 1 \; if \; s_i \to s_j \text{ and } s_j \to s_i, \\ 0 \; otherwise, \end{cases} \tag{6}$$

Those states that have the same rows in Q belong to the same communication class. Algorithm 1 provides pseudocode to determine such communication classes.

Algorithm 1 Determination of communication classes

```
Input: State space $S$ and transition matrix $P$.
Output: Communication classes (C).
1: Check if $s_i \to s_j$ by equation (5) for all $s_i, s_j \in S$.
2: Construct a $k \times k$ matrix $Q$ by equation (6).
3: States that have the same rows in $Q$ belong to the same communication
   class.
```

Definition 3.9. (Closed communication class) A communication class is closed if for all states $s_i \in C$, it holds $\sum_{s_j \in C} p_{s_i s_j} = 1$.

Definition 3.10. (Recurrent class) We say a state is recurrent if whenever the stochastic process leaves the state, the system will always return back in the future. Therefore, for a recurrent state s_i, we have, $P(X_{r+n} = s_i | X_r = s_i) = 1$ for some $n \geq 1$. A class is said to be recurrent if all the states in that class are recurrent. This component also indicates a loop structure, but stronger than the communication class because once customers entered the loop area, they can never escape.

Theorem 3.2. *A finite communication class is recurrent if and only if it is closed.*

Algorithm 2 Determination of recurrent classes

```
Input: Communication classes $C$ and transition matrix $P$.
Output: Recurrent classes (RC).
1: for all $C_j \in C$:
2:     if class $C_j$ is closed (using Definition 3.9):
3:         $C_j$ is a recurrent class (by Theorem 3.2).
```

Algorithm 2 serves the pseudocode to determine recurrent classes. The communication and recurrent classes can also be determined by using reachability analysis (Xie & Beerel, 1998). Furthermore, there are some tools available in MATLAB[2] and R (Spedicato, 2017).

Definition 3.11. (Periodic class) We define the period $d(s_i)$ of a state s_i as the greatest common divisor (gcd) of all n ($n > 0$) that follow $p_{s_i s_i}(n) > 0$. State s_i is periodic if $d(s_i) > 1$. Consider a periodic state s_i with $d(s_i) = 3$, we can only observe it in steps 3, 6, 9 ... when the chain leaves s_i. A class is said to be periodic if all its states are periodic. Therefore, the periodicity component is even stronger than the recurrent property as the DTMC always comes back to the same state in some fixed number of time steps.

Theorem 3.3. *For a communication class, either all states are periodic or none are.*

In practice, for every state s_i, we can construct a set (denoted as D_i) storing all the number of steps the chain traversed back to s_i. Use sequence ‘abcda’ as an example, it takes 4 steps to return ‘a’ and therefore, we append 4 to D_a. If $gcd(D_i) = 1$, state s_i is aperiodic, otherwise it is periodic.

Algorithm 3 Determination of periodic classes

```
Input: State space $S$ and Communication classes (C).
Output: Periodic classes (PC).
1: for all $s_i \in S$:
2:     Construct set $D_i$.
3:     $s_i$ is periodic if $gcd(D_i) > 1$.
4: for all $C_j \in C$:
5:     if all $s_i \in C_j$ are periodic:
6:         $C_j$ is a periodic class (by Theorem 3.3).
```

Definition 3.12. (Absorbing states) An absorbing state is a fixed state that, once reached, the chain will never leave (Norris, 1998). Within the context of business processes, absorption relates to terminating process executions.

[2]https://uk.mathworks.com/help/econ/dtmc.classify.html

4. Proposed Methodology

In this section, we introduce the DTMC-CDD approach and the sliding window framework. Section 4.1 describes the DTMC-CDD approach, its procedures to detect whether there is a concept drift between two event logs. Thereafter, Section 4.2 discusses its application in a process instance stream to detect, localize and rationalize changes using a sliding window framework.

4.1. DTMC-CDD

The problem we encounter is the following: given two event logs referring to process instances within two time windows. Our goal is to inspect whether the process structure or customer behaviour of a process control-flow is significantly different. The solution we propose is to characterize the two logs respectively using seven components originating from DTMCs and then detect differences in such components. Specifically, any changes in the state space (inserting or deleting process executions), communication, recurrent and periodic classes (loop structures), and absorbing states (terminating states) are considered as concept drift. Initial and transition probabilities are mainly used to detect behavioural changes. Algorithm 4 details the components extraction process with an event log as the input.

```
Algorithm 4 Components extraction
Input: An event log $L$.
Output: $S$, $\pi$, $P$, $C$, $RC$, $PC$, $AS$.
 1: Extract the state space, $S = \{s_1, s_2, ..., s_k\}$.
 2: for all $s_i \in S$:
 3:     Calculate the initial probability $\pi_{s_i}$ by using Eq. (2).
 4:     for all $s_j \in S$:
 5:         Calculate the transition probability $p_{s_i s_j}$ by Eq. (3).
 6:     end for
 7: end for
 8: Calculate communication classes ($C$) by Algorithm 1.
 9: Calculate recurrent classes ($RC$) by using Algorithm 2.
10: Calculate periodic classes ($PC$) by using Algorithm 3.
11: Extract absorption states ($AC$).
```

Once we extracted the seven components within two event logs ($L1$ and $L2$), the following process inspects the occurrence of concept drift. To start with, we investigate variations from the process structure perspective (lines 1-6, Algorithm 5). Subsequently, the multinomial test (Read & Cressie, 2012), which is implemented by the XNomial R package[3] in this paper, is used to detect structural and behavioural changes in initial and transition probabilities. We first use the multinomial test to measure the difference between the initial probabilities (lines 7-11). We consider the occurrence of a concept drift if the p-value is smaller than a predefined significance level α (line 7) and then we rationalize if it is a structural or behavioural change. Specifically, it should be considered as a structural change if there exists at least one state $s_i \in S1 \cap S2$ that has the initial probability increases from zero (inserting a start position) or decreases to zero (deleting a start position) (lines 8-9). Otherwise, we argue that the start positions maintain the same but there is a behavioural drift indicating changes in the number of process instances that begin the journal from such initial states.

Thereafter, we apply the multinomial test in two stages for transition probabilities (lines 12-21). The first stage is to check if there is a drift (lines 12-13) and then the second stage to locate which specific state transition leads to the change and whether it is a structural or behavioural change (lines 14-21). Given that there is a difference in the transitions from s_i (line 15), if there exists a state $s_j \in S1 \cap S2$ that has the transition probability increases from zero (inserting a connection, lines 16-17) or decreases to zero (deleting a connection) (lines 18-19), we say it is a structural drift. Otherwise, it is a behavioural drift (lines 20-21). A conclusion is given that there is no concept drift if no significant differences are detected in all the components. Notice that the multinomial test requires two comparing sets having the same size, therefore well suited when S1 equals S2. Otherwise, it is implemented over the states that appear in both S1 and S2.

[3]https://cran.r-project.org/web/packages/XNomial/vignettes/XNomial.html

```
Algorithm 5 Concept drift detection
Input: $S1$, $C1$, $RC1$, $PC1$, $AS1$, $\pi1$, $P1$ and $S2$, $C2$, $RC2$, $PC2$, $AS2$, $\pi2$,
    $P2$. The significance level $\alpha$.
 1: if $S1 \neq S2$:
 2:     Structural changes such as inserting or deleting.
 3: if $C1 \neq C2$ or $RC1 \neq RC2$ or $PC1 \neq PC2$:
 4:     Structural changes in loop structures.
 5: if $AS1 \neq AS2$:
 6:     Structural changes in terminating positions.
 7: if Multinomial test$(\pi1, \pi2) < \alpha$:
 8:     if there is at least one $s_i \in S1 \cap S2$, that have ($\pi1_{s_i} \neq 0$ and $\pi2_{s_i} = 0$)
    or ($\pi1_{s_i} = 0$ and $\pi2_{s_i} \neq 0$):
 9:         Structural changes in the beginning position of $s_i$.
10:     otherwise:
11:         Behavioural changes in beginning positions.
12: if Multinomial test$(P1, P2) < \alpha$:
13:     Changes in state transitions.
14:     for $s_i \in S1 \cap S2$:
15:         if multinomial test$(P1_{s_i}, P2_{s_i}) < \alpha$:
16:             if $p1_{s_i,s_j} = 0$ and $p2_{s_i,s_j} \neq 0$:
17:                 Structural changes (inserting a connection).
18:             if $p1_{s_i,s_j} \neq 0$ and $p2_{s_i,s_j} = 0$:
19:                 Structural changes (removing a connection).
20:             if: $p1_{s_i,s_j} \neq 0$ and $p2_{s_i,s_j} \neq 0$:
21:                 Behavioural changes in transitions from $s_i$.
```

4.2. The sliding window approach

In practice, process instances are usually organized as a data stream because customers enter the business system continuously. In this section, we apply a sliding window approach to partition the process instance stream into a set of populations to detect changes using DTMC-CDD in terms of every consecutive two populations. Those detected change points are then clustered into groups to explore exact locations and also to investigate the reasons for the occurrence.

4.2.1. Change point detection

Sliding window-based techniques can be broadly classified into two categories: fixed-size and adaptive-size, depending on whether the population sizes are required to be fixed. In this paper, a fixed-size sliding window is used, as illustrated in Fig. 2 where P_1 and P_2 refer to successive populations of size w and $x_i, i = 1, 2, ...,$ refers to the recorded process instances. There are two reasons why we use a fixed-size window. First, the positions of detected changes are not required to be very accurate during the detection phase, instead, the precise positions are investigated in the next two stages (change points clustering and localization). Thus, the impact of window size upon the performance of drift detection is decreasing and a fixed window is well suited. Second, it is easier to specify the potential concept drift areas in the following change localization stage.

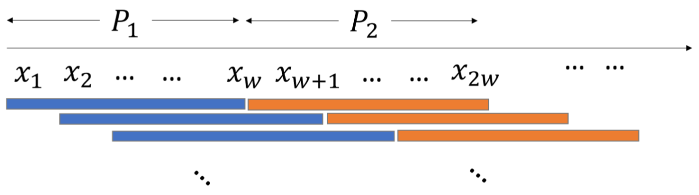


Figure 2: Change point detection

Another particular concern is the selection of a sliding window with or without overlap. By overlapping windows, we mean a part of data belonging to both the populations, i.e., the data at the end of P_1 is also the beginning of P_2. In general, overlapping windows are more sensitive to the occurrence of concept drifts but require more computation resources (Gama et al., 2014). In this paper, we use a non-overlapping window to increase variations between the two populations, which is beneficial for detecting gradual drift (Bose et al., 2013; Maaradji et al., 2017).

4.2.2. Change point clustering

Without loss of generality, around an actual change point, a set of change points can be detected alongside the sliding of the window (as seen in Fig. 3) leading to the issue of pinpointing the exact location of such changes.

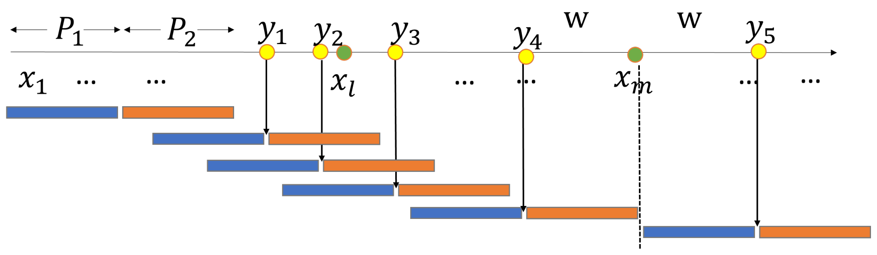


Figure 3: Change point clustering

Assume a perfect concept drift detector with 100% accuracy. Once the actual change point is located between the two populations (P_1 and P_2), a change point can be recorded. As demonstrated in Fig. 3, x_l and x_m are two actual change points, but five change points (y_1, y_2, y_3, y_4 and y_5) have been detected. To address this, a clustering method, DBSCAN (Schubert et al., 2017), was applied to cluster discovered change points into groups. DBSCAN has two main parameters: the maximum distance between two points for one to be regarded as in the neighbourhood of the other (eps) and the minimum number of points required in a neighbourhood to generate a dense region (minPts). We employ the same strategy as (Zheng et al., 2017) to ensure that no candidate change point is dropped by setting minPts = 1. Given window size w, a perfect drift detector and an actual sudden change point, for example, x_m in Fig. 3. With the sliding of the two windows, a change point can be detected after the second window covers x_m (i.e., y_4) and before the first window leaves x_m (i.e., y_5). That is, change points between y_4 and y_5 are detected due to the same actual change point and should be considered in the same neighbourhood of DBSCAN. Therefore, eps is set to $2w$. For gradual drift, we consider changes with the distance less than $2w$ to be inside the same gradual change period.

4.2.3. Change point localization

Once the detected change points have been clustered into groups, the exact change point positions are investigated in every cluster. This can be seen in the above-mentioned example where three detected change points y_1, y_2, y_3 are clustered in one group, as presented in Fig. 4. Given detected change points in one cluster, Algorithm 6 describes the procedures to explore change positions. The first step is to find the minimum and maximum index of change point (line 1) to determine the concept drift area, i.e., from x_{lower}

to x_{upper}, denoted as $[x_L, x_U]$ (line 2). Thereafter, the process drift region is split into three subpopulations of the same length (line 3) where the DTMC-CDD approach is applied in the left (P_1' and P_2') and right (P_2' and P_3') subdomains, respectively (line 4). The region splitting process is continued to locate change points in a smaller area until the number of process instances in the area is smaller than a predefined value β (line 5-8). As discussed above, changes in a range of $2w$ can be generated due to the same drift, in this paper, we set β to 10% of $2w$, i.e., $0.2w$. Here x_{left} and x_{right} refer to the start and end of P_2'.

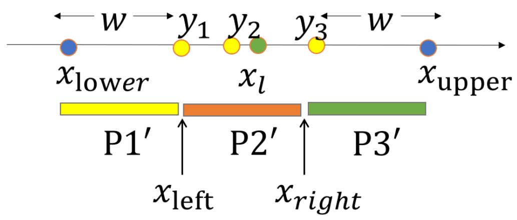


Figure 4: Change point localization

Algorithm 6 Change point localization

```
Input: Detected change points $Y = \{y_1, y_2, ..., y_n\}$, population size $w$ and
    terminating parameter $\beta$.
1: Get $y_{min} = min(Y)$ and $y_{max} = max(Y)$.
2: Get the concept drift area,
       $x_L = y_{min}$-$w$ and $x_U = y_{max}$+$w$.
3: Split process instances under $[x_L, x_U]$ into 3 halves, $P_1'$, $P_2'$, and $P_3'$ as
   illustrated in Fig. 4.
4: Apply DTMC-CDD on the left ($P_1'$ and $P_2'$) and right ($P_2'$ and $P_3'$) sub-
   population, respectively.
5: if concept drifts occurred between $P_1'$ and $P_2'$:
6:     Update $x_U = x_{right}$, return $x_{left}$ as the exact change
       position when $x_U$-$x_L \leq \beta$, otherwise, goto Step 3.
7: if concept drifts occurred between $P_2'$ and $P_3'$:
8:     Update $x_L = x_{left}$, return $x_{right}$ when $x_U$-$x_L \leq \beta$,
       otherwise, goto Step 3.
```

For sudden drift, change localization aims to investigate the exact point of the occurrence. However, in a scenario of gradual drift, the previous business process is replaced with a new process model gradually. Therefore, the goal is to discover the start and end point of the variation, which is a challenging task as changes are usually not significant enough during such periods (Martjushev et al., 2015).

4.2.4. Change point rationalization

Once the exact change points are discovered, the DTMC-CDD approach is applied repeatedly for every change point to investigate the reasons leading to the drift. More specifically, to uncover if a process drift happened because of changes in process structure (the insertion or deletion of process executions, changes in the beginning or terminating states and loop structures) or in customer behaviour (changes in behavioural patterns).

5. Evaluation on artificial event logs

In this section, we first describe evaluation metrics used in this paper within the context of sudden and gradual drifts. Thereafter, a discussion is given for experiments setup, especially the specification of the window size. Following that, we evaluate the performance of the DTMC-CDD method under 52 artificial logs from a loan application. Specifically, the 18 event logs for sudden drifts are from (Maaradji et al., 2017), then we created another 18 logs referring to gradual changes. Furthermore, based on the base model (presented in Fig. 1 a), we generated 8 event logs each representing behavioural drift that occurred suddenly or gradually. Finally, the proposed method is compared with three commonly used approaches: relation type count (RTC) proposed by Bose et al. (2013), relation entropy (RE) proposed by Bose et al. (2013) and partial order runs (RUN) proposed by Maaradji et al. (2017).

5.1. Evaluation metrics

Precision, recall and F1-score are frequently used performance measures in classification problems (Tharwat, 2021). In some scenarios, precision is more important than recall and vice versa. For example, in a cancer detection problem, recall is more important because the wrong classification (false negatives, i.e., wrongly considering a cancer patient as non-cancer) is unacceptable. On the other hand, precision is more important for scenarios where we are only interested in false positives. In a concept drift detection context, it is insufficient to detect both false positives (i.e., considering non-drifts as changes) and false negatives (i.e., considering real changes as non-drifts). As a result, the F1-score, which is the harmonic mean of precision and recall, is used as the performance measure.

In order to use the F1-score to evaluate the accuracy of the detected change points within the context of streaming data, we employ an approach proposed by Martjushev et al. (2015). Here, a lag period (l) around the true drift points is established whereby a change point is considered a true positive (TP) if it locates a change point within the lagged region of a real change point (as seen in Fig. 5). The green and yellow circles refer to a real change point and a detected change point, respectively.

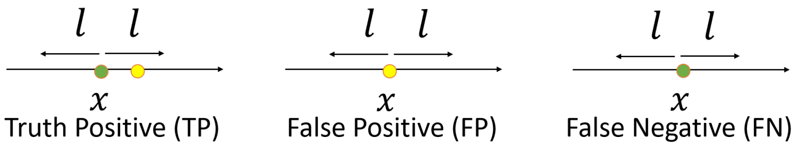


Figure 5: Modified F1-score for sudden drifts

Specifically, the true positive (TP), false positive (FP) and false negative (FN) are adjusted as, TP: a real change point at x and there is a detected drift point within $x \pm l$. FP: a detected change point at x, but there is no real drift within $x \pm l$. FN: a real drift occurred at x, but no change points were detected within $x \pm l$. By using TP, FP and FN, the precision, recall and F1-score can be calculated as,

$$precision = \frac{TP}{TP + FP}. \tag{7}$$

$$recall = \frac{TP}{TP + FN}. \tag{8}$$

$$F1\text{-}score = 2 * \frac{precision * recall}{precision + recall}. \tag{9}$$

The F1-score in this format is suitable for sudden drifts. Considering gradual drifts, however, the real changes are located in a continuous period rather than a specific point. Therefore, a detected change point is considered as a TP if it locates in any one of the real change regions, as seen in Fig. 6

where the start and end of the gradual drift are x_1 and x_2 respectively. In other words, the green bar refers to the drift area and the yellow circle is the detected change point.

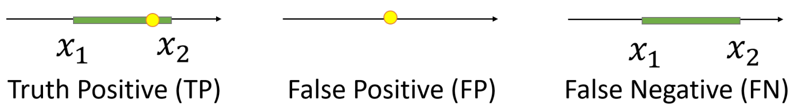


Figure 6: Modified F1-score for gradual drifts

5.2. The specification of window size and stride

The specification of window size is critical and challenging in any sliding window-based methods (Maaradji et al., 2017) since a small population size usually leads to false positives, i.e., detecting concept drifts that do not exist (Martjushev et al., 2015), while a large one might reduce the variability of process instances, resulting in the missing of change points (Maaradji et al., 2017). In the context of supervised learning, the window size can be optimized according to the performance of the concept drift detector (Gama et al., 2014). There is, however, a general issue for unsupervised learning (Žliobaitė et al., 2016; Gama et al., 2014). To address the window size, we implement DTMC-CDD, RTC, RE and RUN under different w with the size ranging from 30 to 100 in increments of 10 under all the 52 event logs. Their best performance with the highest F1-score under such window sizes are then compared.

The two sliding windows, in many scenarios, are moving one step every time to detect changes for every process instance (Bose et al., 2013; Maaradji et al., 2017; Martjushev et al., 2015; Seeliger et al., 2017). However, it is computationally time-consuming and lots of redundant change points might be detected. To accelerate the drift detection process and reduce the number of candidate change points, in this paper, the stride (i.e., the number of steps the window slides every time) is set as 50% of the window size as a trade-off between the time consuming and sensitiveness to changes.

5.3. Experiments on sudden drifts

5.3.1. Scenario 1. Sudden drifts in process structure

As mentioned in Maaradji et al. (2017), 18 alternative models, compared to a base model (demonstrated in Fig. 1 a) of a loan assessing application were generated for simulating sudden drifts. To construct a process instance stream involving a set of change points, five logs of 250 instances from the base and one of the altered models were generated, respectively and thereafter they were combined as a data stream (Maaradji et al., 2017). The stream generation process was continued until all the altered models were used and finally, 18 data streams were created. An example is shown in Fig. 7 using the control-flow of (a) and (b), as presented in Fig. 1.

The blue and red boxes refer to the base and the altered model, respectively leading to 9 real changes at locations 250, 500, 750, 1000, 1250, 1500, 1750, 2000, and 2250. By applying the DTMC-CDD approach, the detected changes are located at positions 225, 480, 750, 990, 1230, 1500, 1740, 1980 and 2235. As an example, given the lag level of F1-score as 10, the number of true positives (TP) is 4, indicating 4 of 9 detected change point locates within 10 steps of an exact drift point, leading to the F1-score of 0.45. Furthermore, increasing the lag region to 20, the F1-score goes up to 0.89. Besides, the p-values of the multinomial test using transition probabilities are plotted in Fig. 8, indicating nine change points with a p-value below 0.05. The F1-scores of the DTMC-CDD method under the 18 types of concept drift are shown in Fig. 9.

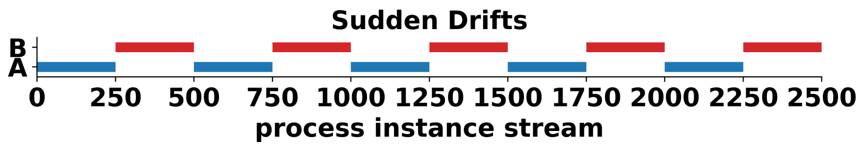


Figure 7: Log generation for sudden drifts in process structure

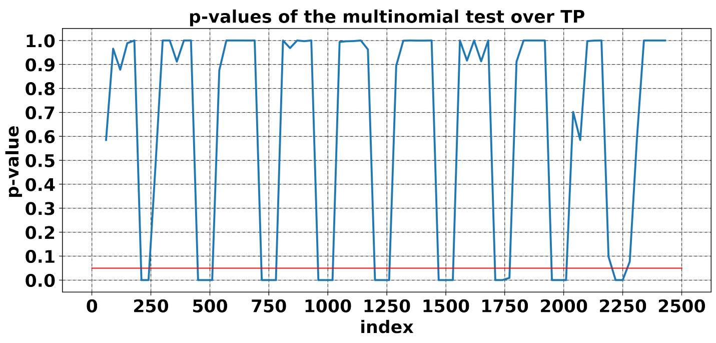


Figure 8: The multinomial test over transition probabilities (sudden drift, control-flow 'cf')

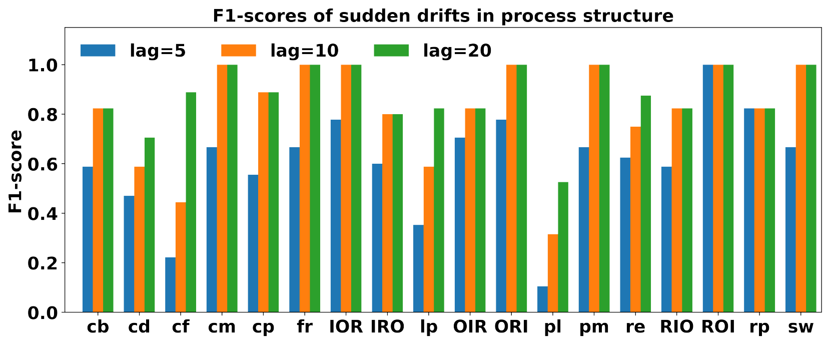


Figure 9: F1-scores of sudden drifts in process structure

According to Fig. 9, the F1-score is above 0.7 for most of the event logs, excluding 'pl' (0.526) under lag region 20. Compared with the base and the 'pl' model, two process executions: 'Check credit history' and 'Assess loan risk' change to be sequential from parallel, however, there are no significant changes in their related transition probabilities, leading to the ineffectiveness of the process drift detection.

5.3.2. Scenario 2. Sudden drifts in customer behaviour

To construct a process stream involving sudden changes in customer behaviour, we modified some transition probabilities in the base model (Fig. 1, a). In the original base model, the transition probability from 'Check application form completeness (CAC)' to 'Return application back to applicant

(RAA)' and 'Check credit history (CCH)' are 0.097 and 0.903, respectively, indicating that nearly 10% customers did not submit the application form correctly and customers property was always assessed ahead in the credit history. In order to generate sudden changes in customer behaviour, we created eight logs by increasing the transition probability from 'CAC' to 'RAA' gradually from 0.2 to 0.9 with an increment of 0.1. It can be considered as a scenario where an applicant is required to be assessed more strictly and also, the application is harder to be approved. Furthermore, the transition probability to reject and cancel the application was increased to 0.7 and 0.8 from 0.5 and 0.5, respectively.

Thereafter, process instances from the original base model and the newly generated logs were combined to construct the process stream. As seen in Fig. 10, the blue bars refer to process instances from the original base model, while our simulated process instances are represented by red boxes. Different from the strategy in Fig. 7, all process instances follow the same process structure (Fig. 1 a), but with customer executions or preferences changed. Evaluated by the multinomial test, the F1-scores are demonstrated in Figure 11. The numbers 1-8 on the horizontal axis correspond to the transition probabilities from 'CAC' to 'RAA' in our created logs which vary from 0.2 to 0.9. The F1-score is unsatisfactory with a value of 0.45 when the transition probability is 0.2. On the other hand, the F1-score is over 0.8 under the remaining seven scenarios.

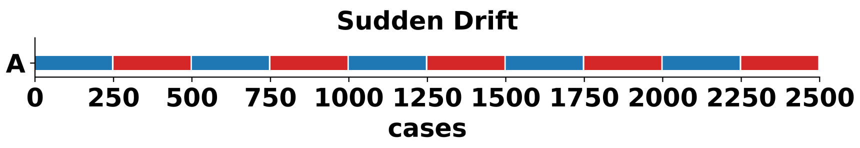


Figure 10: Log generation for sudden drifts in customer behaviour

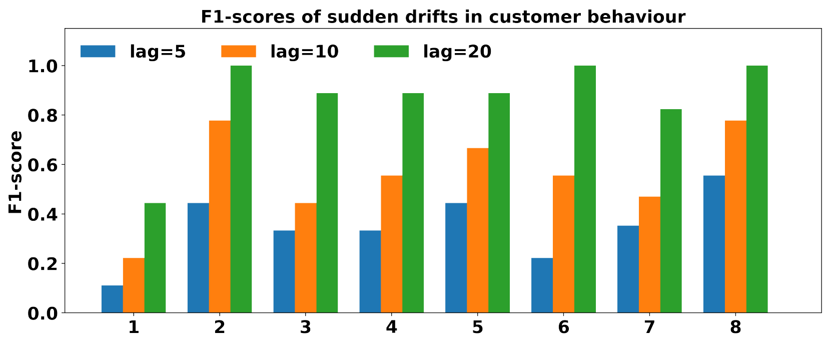


Figure 11: F1-scores of sudden drifts in customer behaviour

5.4. *Experiments on gradual drifts*

5.4.1. *Scenario 1. Gradual drifts in process structure*

Within the context of gradual drift in the process structure, the number of process instances in the current process model decreased gradually while the other new model takes over. Fig. 12 demonstrates a process instance stream (involving 2000 cases) under gradual changes. Specifically, the blue and red boxes refer to the base and altered model, respectively, the same as in sudden drift. During the period of gradual changes (the green and orange boxes), instances from both models coexist.

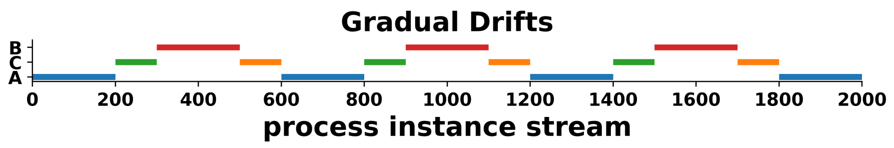


Figure 12: Log generation for gradual drifts in process structure

Bose et al. (2013) consider gradual concept drift as a linear variation. In other words, the fading of one process model and the taking over of the new model happen linearly. In this paper, we applied the same strategy to generate process instance streams involving gradual changes. As illustrated in Fig. 13, the x-axis refers to the 100 process instances in the green and orange boxes in Fig. 12 and the y-axis relates to the probability density function determining where a specific process instance is sampled (i.e., from model A or B). We take the first green bar as an example, referring to instances from 200 to 300 where process model A is gradually replaced by B. The dashed blue and solid red line represent the probability of sampling an instance from model A and B, respectively. Therefore, process instances before the crossover point (0-50) are more likely to be sampled from A, and vice versa for instances after the crossover point. The experimental setup for the orange bars is represented in Fig. 13 (b). Consequently, 18 types of gradual drifts were generated.

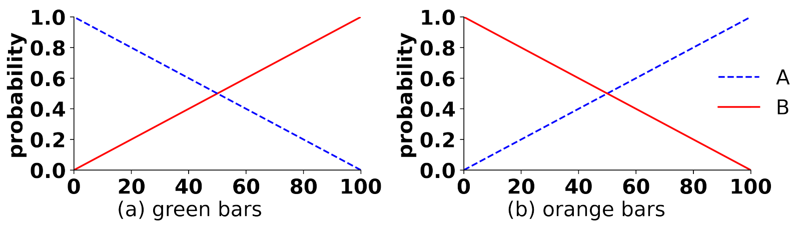


Figure 13: Probability of sampling during the periods of gradual drift

The performance of the DTMC-CDD approach under such 18 gradual drifts was evaluated by using F1-score, as seen in Fig. 14. The performance was satisfactory under the majority of event logs with the F1-score over 0.8.

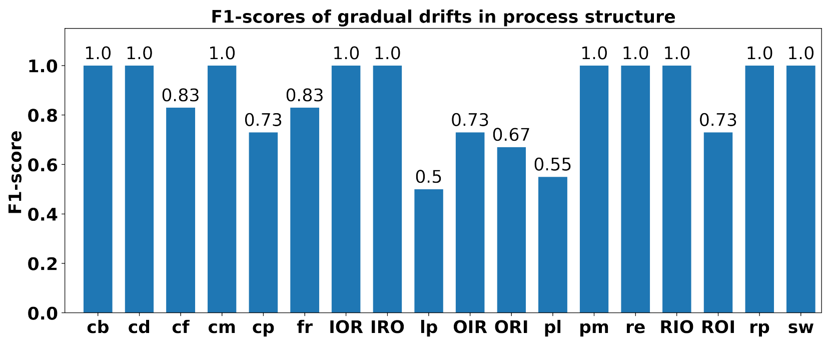


Figure 14: F1-scores of gradual drifts in process structure

5.4.2. Scenario 2. Gradual changes in customer behaviour

Process instances from the original base model and the modified base model referring to changes in customer behaviour (as mentioned in Section

5.3.2) were used to construct the process instance stream involving gradual changes resulting in the F1-score of 0.918 on average.

5.5. Method comparison

In this section, we compared our method with three commonly used approaches: relation type count (RTC) (Bose et al., 2013), relation entropy (RE) (Bose et al., 2013) and partial order runs (RUN) (Maaradji et al., 2017) under the above-mentioned 52 event logs.

5.5.1. The follows/precedes relation

RTC and RE are based on the follows/precedes relation, i.e., for process execution s_i and s_j, if s_i always, sometimes or never follows s_j (Bose et al., 2013). Recalling the base model of the loan application, mentioned in Fig. 1 (a). For example, execution 'Appraise property (AP)' always follows 'Check application form completeness', 'Receive updated application' and 'Return application back to applicant', meanwhile never follows the other states. Therefore, totally there are 3 states that 'AP' always follows behind, and 12 states that 'AP' never follows. Thus, we have $RTC(AP) = \langle 3, 0, 12\rangle$. RE is the entropy of the RTC metric, which is calculated by $RE = -p_A \times log_2(p_A) - p_S \times log_2(p_S) - p_N \times log_2(p_N)$ where p_A, p_S and p_N represent the probability that a specific state always, sometimes and never follows the others. therefore, $RE(AP) = 0.722$.

RTC and RE are well suited for structural changes of control-flow but insensitive to concept drifts in customer behaviour. Referring to the example in Section 5.3.2, the transition probability from 'Check application form completeness' to 'Return application back to applicant' is increased from 0.097 to up to 0.9, indicating a concept drift in customer behaviour. However, the process structure maintains unchanged leading to the result that the RTC and RE feature are the same under the two processes. Consequently, RTC and RE fail to detect the changes. For the proposed DTMC-CDD method, concept drifts in customer behaviour can be detected by the changes in transition probabilities.

RE features, in some scenarios, have the same value but actually refer to totally different business models. Consider a scenario where the RTC metric changed to $\langle 2, 9, 3\rangle$ from $\langle 9, 3, 2\rangle$, indicating a concept drift, while the RE value is the same leading to the failure of the detection. Consider another alternative version ('cb'), demonstrated in Fig. 15 where a direct link is added comparing with the base model. However, evaluated by RTC and RE, there is no difference at all as the follows relation remains the same leading to the failure of change points detection.

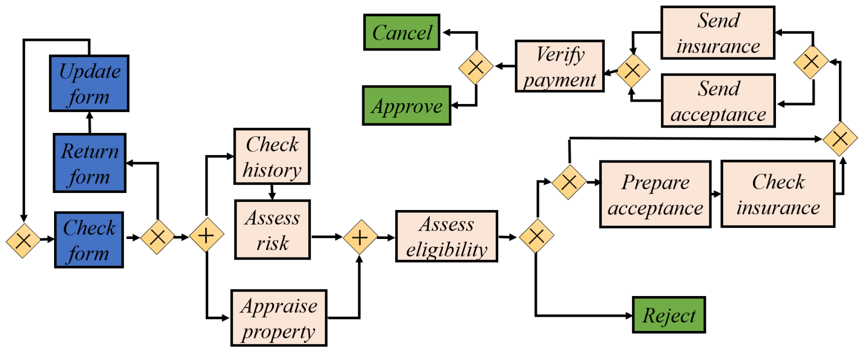


Figure 15: Control-flow ‘cb’ of the loan application process

5.5.2. The RUN relation

Maaradji et al. (2017) proposed a method, namely RUN to detect process concept drift. It is defined as a feature vector, representing the number of process instances in different customer pathways or journeys. Customers following the same pathways excluding the concurrency structure (paralleled executions) are considered in the same RUN. Consider a set of process instances following three unique pathways: ‘abcd’ (10 customers), ‘acbd’ (5 customers) and ‘aedf’ (8 customers) where execution ‘b’ and ‘c’ are concurrency executions. Therefore, customers under ‘abcd’ and ‘acbd’ are considered in the same RUN ‘a(bc)d’ with 15 customers leading to a feature vector as $\langle 15, 8 \rangle$.

Due to the considering of pathways, the feature of RUN can work well for detecting concept drifts in customer behaviour but may fail for behavioural changes with parallel executions. For example, the number of customers under ‘abcd’ and ‘acbd’ changes to 1 and 14 from 10 and 5, indicating more customers prefer to execute ‘c’ before ‘b’. However, the feature vector still is $\langle 15, 8 \rangle$. Another issue is the impact of loop structures, leading to uncountable unique pathways. In DTMC-CDD, communication, recurrent and periodic classes are used to capture loop structures and therefore, might perform more robustly.

5.5.3. Method comparison under the F1-score

The performance of DTMC-CDD, RTC, RE and RUN was compared under the above-mentioned 52 logs involving sudden and gradual drifts in both process structure and customer behaviour of control-flow. The lag level was specified as 20. Furthermore, statistical test methods in line with those used in the original publications for RTC, RE and RUN were applied. Specifically, the Hotelling T^2 hypothesis test was applied for RTC features (Bose et al., 2013) while for RE and RUN, the Kolmogorov-Smirnov test (KS test) (Bose et al., 2013; Sheskin, 2003) and the Chi-square test were used (Maaradji et al., 2017; Nuzzo, 2014), respectively. For the proposed DTMC-CDD approach, the multinomial test was applied to evaluate the significance of initial and transition probabilities. The significance level is set as 0.05 for all the approaches.

Table 2: The F1-scores of structural changes

	Sudden Drifts				Gradual Drifts			
	RTC	RE	RUN	DTMC	RTC	RE	RUN	DTMC
cb	0.235	0.133	**1.0**	0.824	0.182	0.5	0.769	**1.0**
cd	**1.0**	0.889	0.889	0.706	0.923	0.857	0.25	**1.0**
cf	0.2	0.167	0.778	**0.889**	0.4	0.308	0.727	**0.833**
cm	0.353	0.375	0.947	**1.0**	0.2	0.308	**1.0**	**1.0**
cp	**1.0**	0.737	**1.0**	0.889	0.364	0.833	**0.923**	0.727
fr	0.25	0.4	0.947	**1.0**	0.182	0.462	**0.923**	0.833
IOR	**1.0**	0.588	0.889	**1.0**	0.308	0.375	**1.0**	**1.0**
IRO	**1.0**	0.889	0.353	0.8	0.4	0.667	0.727	**1.0**
lp	0	0.143	0.556	**0.824**	0.2	**0.5**	0.4	**0.5**
OIR	**1.0**	0.889	**1.0**	0.824	0.444	0.429	**0.833**	0.727
ORI	**1.0**	0.824	0.778	**1.0**	0	0.462	**1.0**	0.667
pl	0	0.167	0	**0.526**	0.2	0.364	**0.6**	0.545
pm	0.588	0.889	0.889	**1.0**	0.2	0.706	**1.0**	**1.0**
re	0.462	0.667	0.737	**0.875**	0.222	0.667	0.857	**1.0**
RIO	0	0.778	**1.0**	0.824	0.444	0.5	0.923	**1.0**
ROI	0.889	0.842	**1.0**	**1.0**	0	0.625	**0.923**	0.727
rp	**1.0**	0.625	**1.0**	0.824	0.571	0.471	0.923	**1.0**
sw	0.5	0.889	**1.0**	**1.0**	0.333	**1.0**	**1.0**	**1.0**
avg	0.582	0.605	0.82	**0.878**	0.31	0.557	0.821	**0.864**

Table 3: The F1-scores of behavioural changes

	Sudden Drifts				Gradual Drifts			
	RTC	RE	RUN	DTMC	RTC	RE	RUN	DTMC
1	0.154	0.182	**0.471**	0.444	0.2	0.364	0.364	**0.923**
2	0	0.222	0.824	**1.0**	0	0.182	0.545	**0.833**
3	0.154	0.133	0.667	**0.889**	0	0.182	0.6	**0.833**
4	0	0.211	0.778	**0.889**	0.222	0.364	0.545	**0.923**
5	0	0.118	0.667	**0.889**	0.25	0.25	0.833	**1.0**
6	0	0	0.667	**1.0**	0	0	0.6	**0.833**
7	0	0.111	**1.0**	0.824	0.25	0.25	0.833	**1.0**
8	0	0.125	0.706	**1.0**	0	0	0.909	**1.0**
avg	0.038	0.138	0.722	**0.867**	0.115	0.199	0.654	**0.918**

Table 2 illustrates the simulation results where the first column refers to the 18 types of structural changes. Every row states the F1-score of the four compared methods under sudden and gradual drifts. The averaged F1-score of each approach is depicted in the last row. In general, the RUN and DTMC-CDD method perform much better than RTC and RE and the DTMC-CDD approach outperforms RUN in the averaged F1-score. Comparing RTC and RE, RE outperforms RTC in almost all event logs. One of the possible reasons is that RE involves fewer features and therefore, is more likely to be considered as significant by using statistical tests. A similar result is given on the detection of behavioural changes, as demonstrated in Table 3. Notice that the performance of RTC and RE decreases dramatically with an F1-score of zero under the majority of event streams.

The experiments were carried out on Windows10 64-bit operating system, 16GB memory, Intel(R) Core(TM) i5-8250U CPU. There is, in general, no significant difference in the time consuming (seconds) of RTC (31.4), RE (31.7) and RUN (36.3). While it requires much more time for our proposed DTMC-CDD method (56.3) as it involves multi-components required to be calculated and multi-statistic tests for understanding the reasons for the changes. Specifically, it requires similar computational resources as other approaches if using transition probabilities as the only feature. In other words, the root cause for the additional time spent is on the calculation of communication, recurrent and periodic classes.

6. Evaluation on a real-life data set

In this section, we validate the proposed DTMC-CDD method in a real-life business process event log referring to a hospital billing (HB) process to detect, localize and rationalize concept drifts.

6.1. Data collection

The event log refers to a billing process for medical services in a hospital in the Netherlands. For a specific patient, when a particular period of time has passed or the treatment is finished, a billing package containing a number of medical services (such as medical diagnostics, hospitalisation, treatment and surgeries) is billed together (Mannhardt, 2018).

The data used in this section was collected between December 2012 and January 2016, involving 18 unique process executions, 451359 events and 100000 cases (Mannhardt et al., 2017). Because changes in the laws and regulations of medical services every year, there is a large number of special cases and mistakes when selecting the correct billing package and declaration code. Consequently, the billing package can be rejected, canceled or reopened in some cases. In addition, these changes lead to a non-stationary billing environment, addressing the opportunity for process drift analysis.

6.2. Case selection

The HB event log comprises 11 main activities, covering 99% of the observed process instances (98515), as presented in Fig. 16.

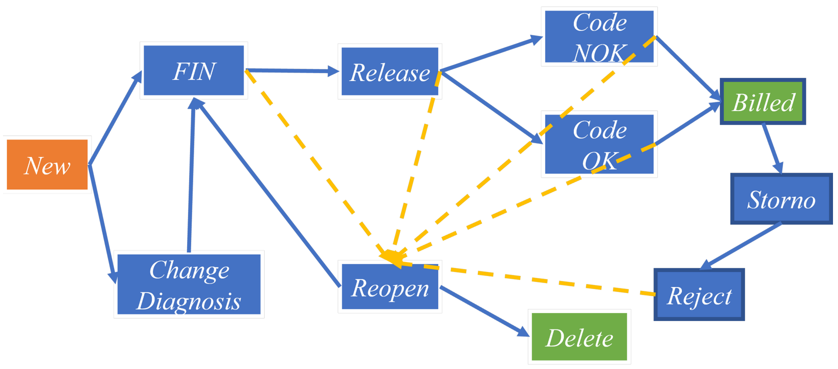


Figure 16: The diagram of the hospital billing event log

Process instances involving the other 7 activities were considered as outliers and excluded from the data stream. In general, all patients start their billing process from state 'New' and then the billing process moves to state 'FIN' directly or after changing the diagnosis. Thereafter, the code requires to be inspected and then, the process is terminated in 'Billed' or 'Deleted'. Table 4 describes the meaning of these process activities.

Table 4: Main activities in the HB event log (Mannhardt, 2018)

Activity	Description
New	The creation of a new billing package.
FIN	The billing package is completed and can no longer be changed.
Release	The billing package is released for delivery to the insurance company.
CodeOK	The obtaining of a correct declaration code.
Billed	The billing package has been billed, with an invoice that is sent out.
Change Diagnosis	The change of the diagnosis.
Delete	The deletion of the billing package.
Reopen	The reopening of the billing package, with the goal of introducing additional medical services or removing existing services.
CodeNOK	The obtaining of a declaration code with an error message.
Storno	The cancellation of a billing package.
Reject	The rejection of the invoice that is sent to the insurance company.

6.3. Experimental design

Since there is no ground truth for concept drift points provided in the HB event log, evaluation metrics such as F1-score cannot be used, leading to difficulties in the specification of the window size and stride. To address this, a set of simulations were executed based on a sample of the whole event log (the first half-year, i.e., process instances beginning from 16th, December 2012 to 1st, July 2013, involving 18173 instances) under different window sizes (500-2500) and strides (200-1000). From this sample, the best model was determined with a corresponding window size of 1100 and a stride of 400. The resulting model was then applied to the whole event log.

As presented in Fig. 17, the blue solid line relates to the p-values of the multinomial test under a transition matrix with the significance level 0.05 (red solid line). The green and black (only in subfigure a) points refer to changes in the communication class and recurrent state, respectively. There are no changes detected in the state space, periodic class, absorbing states and initial probabilities. Note that the x-axis represents the entering order of patients, but they were reduced by 1000 times. More specifically, the index from 0 to 18 of the x-axis actually means index from 0 to 18000.

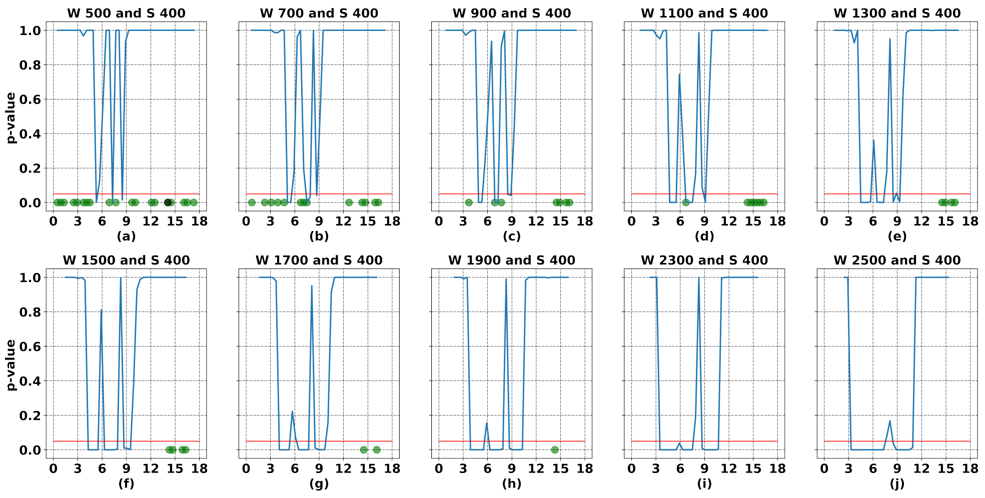


Figure 17: DTMC-CDD under window size from 500 to 2500 with fixed stride 400

According to Fig. 17, there are three concept drift areas around index 5100, 6300 and 8700 because the transition probabilities are lower than the significance level of 0.05. Taking index 5100 as an example, the transition probability from 'New' to 'Change Diagnosis' decreased from 0.559 to 0.368, meanwhile the probability of staying at 'New' increased to 0.306 from 0.014, indicating changes in customer behaviour. One of the possible reasons is that the quality of the hospital service declined, resulting in a delay in the billing process. Alternatively, patient behaviour in 6300 is the opposite of that at 5100 where the transition probability from 'New' to 'Change Diagnosis' increased to 0.538 from 0.361 and the probability of staying decreased to 0.052 from 0.3, possibly indicating the recovery of the billing environment. The occurrence of changes in 8700 is because of the transitions of execution 'New', which is similar to change point 6300.

With the increase of the window size, the concept drift areas become larger, indicating a decrease in sensitivity (as presented in Fig. 17 i and j). On the other hand, under small window sizes (Fig. 17 a, b and c), many false positives (i.e., the green points) were detected. Such green points referring to changes in the communication class, but actually they are not real concept drifts. According to further analysis, all of the green points are related to the execution of 'Change Diagnosis'. Taking index 900 as an example, the transition probability from 'Reopen' to 'Change Diagnosis' decreased to 0 from 0.02 leading to the fact that it cannot communicate with other states, indicating a concept drift in control-flow. However, the control-flow of the process model remains unchanged and such false detections result from infrequent transitions.

To reduce the number of false positives and maintain the sensitivity to changes, window sizes from 500-900 and 2300-2500 are not considered. Furthermore, to avoid overfitting (i.e., the model fits the training data perfectly leading to the issues in the generalization), window sizes 1700 and 1900 are not considered as almost all false positives are filtered. Therefore, we noted that window sizes 1100, 1300 and 1500 are reasonable and 1100 was selected in the following simulations. For stride, as presented in Fig. 18, under 1000, the concept drifts around 4500 and 6500 can be detected correctly, but only at one point. Actually, according to the data, concept drift near such areas should be considered the gradual type. Therefore, the stride of 800 and 1000 are not satisfactory. The p-values under stride 200 are fluctuating a lot and so, it is not a good option. Overall, we consider strides 400 and 600 to be satisfactory and, in this paper, we applied 400 for the following simulations.

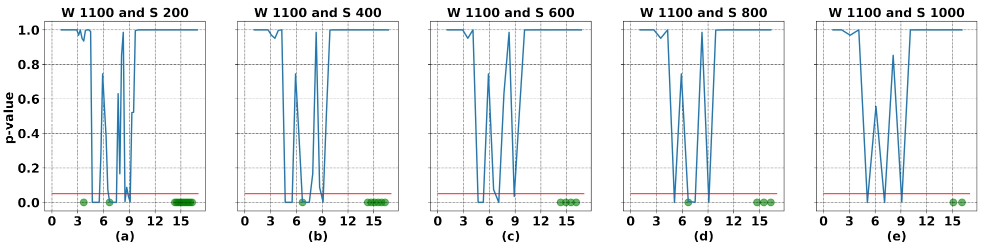


Figure 18: DTMC-CDD under stride from 200 to 1000 with fixed window size 1100

6.4. Analysis

In this section, we use a sliding window of size 1100 and stride 400 to detect, localize and rationalize concept drift of the whole HB event log.

6.4.1. Concept drift detection

As demonstrated in Fig. 19, the blue line represents the p-values of the multinomial test, indicating changes in transition probabilities. The horizontal axis relates to process instances in chronological order. By comparing transition probabilities, six sudden drifts are detected. These drift points are labelled as A-F, located at the index of 4700, 5900, 8700, 16300, 60700 and 70300. In addition, there are 4 (red circles in Fig. 19), 50 (green circles), 3 (black circles) and 1 (orange circles) change points that are detected by comparing the state space, communication class, recurrent class and absorbing states, respectively. There are no changes in periodic class and initial probabilities. In summary, 64 change points are detected in the drift detection stage.

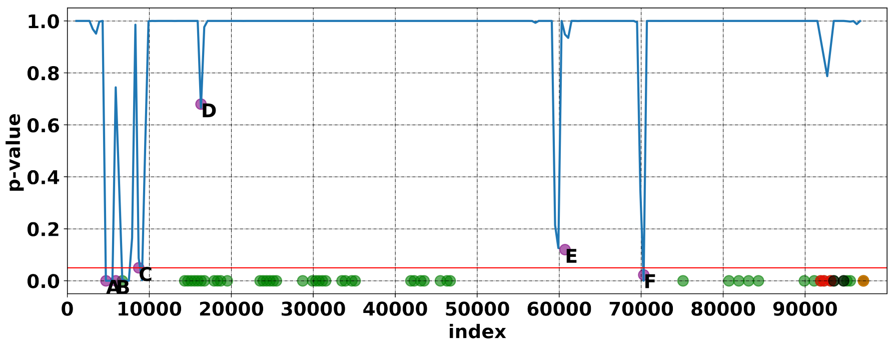


Figure 19: Drift detection in the HB event log

6.4.2. Concept drift localization

In the localization stage, detected change points are clustered into groups to investigate their exact locations, as illustrated in Fig. 20. For change points due to variations in transition probabilities, their exact locations are 4100, 6500, 8800, 16000, 60000 and 70000, nearly the same as the results from the detection stage. However, several change points detected by the state space, communication and recurrent class, are merged or filtered, leading to a decrease in the number of changes (i.e., 2, 23, 1 and 1).

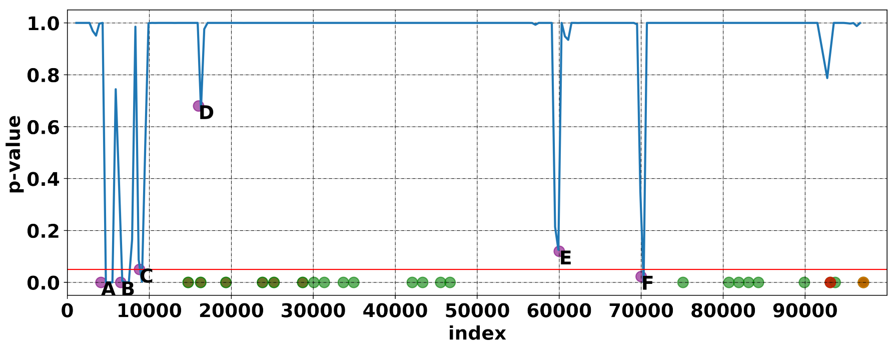


Figure 20: Drift localization in the HB event log

6.4.3. Concept drift rationalization

According to further analysis, all of the green points (changes in communication classes) are false positives due to infrequent transitions between ‘Reopen’ and ‘Change Diagnosis’. Taking index 14733 as an example, the transition probability from ‘Reopen’ to ‘Change Diagnosis’ decreased to 0 from 0.02 leading to the fact that it cannot communicate with other states, indicating a concept drift in the (loop) structure of control-flow. However, the process model remains unchanged and such false detections result from infrequent transitions. On the other hand, for change point 93066, the changes occurred in the state space where no applicants were rejected in the population before 93066, indicating changes in the structure of the billing process. The occurrence of drift at point 97100 is due to the unexpected absorbing states. Specifically, some patients terminated their billing process at execution ‘Deleted’ instead of ‘Billed’.

Regarding change point A, the transition probability from ‘New’ to ‘Change Diagnosis’ decreased from 0.559 to 0.368, meanwhile the probability of staying at ‘New’ increased to 0.306 from 0.014, indicting changes in customer behaviour. One of the possible reasons is that the quality of the hospital service declined, resulting in the delay of the billing process. Alternatively, patient behaviour in point B is the opposite of that at A where the transition probability from ‘New’ to ‘Change Diagnosis’ increased to 0.538 from 0.361 and the probability of staying decreased to 0.052 from 0.3, possibly indicating the recovery of the billing environment. The occurrence of changes in point C is because of the transitions of execution ‘New’, which is similar to change point A. As for instance D, even though the p-value is around 0.7, much bigger than the significance level 0.05, it still should be considered as a concept drift because it is significantly smaller than non-drift points. Particularly, the probability for a patient staying at 'New' was declined to zero, indicating an improvement in the billing process. The transition probability from 'Code Nok' to other states is significantly different before and after point E. More specifically, the probability from 'Code Nok' to 'Billed' decreased from 0.489 to 0.205, while from 'Code Nok' to 'Code Ok', it increased to 0.716 from 0.133. Concept drift in point F is due to the transitions from 'Storno'. The transition probability from 'Storno' to 'Billed' increased to 0.875 from 0.184, while from 'Storno' to 'Reject', it declined from 0.816 to 0.125, representing a sharp decrease in the number of successful billings, impacting negatively on the hospital billing environment.

6.5. Lessons learned

So far, we have addressed the collected data, selected cases, designed experiments and experimental results. We then discuss several findings and the lessons we learned. To start with, the window size and stride have a considerable impact on the DTMC-CDD method. Specifically, large values can diminish the difference between populations, resulting in drifts being undetected. A small window and stride size, on the other hand, may reveal redundant change points, causing difficulties in focusing on main drifts. As a result, it is an acceptable strategy to optimize/tune the two parameters based on a sample of the whole data, especially under an unsupervised learning scenario.

Second, the transition probability, as one of the seven components in the DTMC-CDD method, plays an important role in detecting behavioural drifts. In addition, it can also indicate structural changes, such as the removal of a connection between two activities, in which case the transition probability will be zero. Third, the communication, recurrent and periodic classes can discover loop structures with different characteristics. In some scenarios, however, the research focus is on detecting drifts in loop structures only. In such a case, using merely the communication class is sufficient and efficient.

7. Discussion

By evaluating the proposed DTMC-CDD approach on artificial and real-world datasets, we highlight the advantages and disadvantages of the proposed approach.

The first advantage is that the method provides a single model for detecting, localizing and rationalizing sudden and gradual, as well as structural and behavioural concept drifts. By introducing the initial and transition probability from DTMCs, it can handle behavioural drifts at an event-level, i.e., addressing changes in the proportion of a specific event pair. However, the majority of methods in the literature concentrate on changes in the case-level, i.e., the proportion of different traces/pathways. As a result, the DTMC-CDD method can deeply investigate the root causes of changes. The ability for handling gradual drifts is due to the use of a non-overlapping sliding window that can increase differences in populations to highlight gradual changes.

Second, the approach has no hyper-parameters. As a result, it requires no prior knowledge and is easy to implement in various applications. In general, it is time consuming and challenging for hyper-parameter tuning/optimization, especially in unsupervised learning scenarios because the performance is difficult to validate compared to supervised methods. In our sliding window framework, on the other hand, there are two main hyper-parameters: the window and stride size, but they almost exist in any sliding window methods. We recommend determining them on a regular basis, such as weekly or monthly. Otherwise, they can be fine-tuned manually using a sub-event log and then applied to the rest of the log.

According to experiments on the HB event log, DTMC-CDD can detect behavioural drifts well but it also discovers numerous false positives in the process structure, such as changes in the state space. We say they are false positives because they are detected as drifts but in fact, the process maintains the same. These false positives are commonly caused by infrequent process transitions. A straightforward solution is using a frequency threshold to ignore the impact of infrequent cases, but it also introduces a new hyper-parameter, which is not what we expected.

In addition, the change localization process is time-consuming because it iteratively explores variations. However, in some scenarios, it is not necessary to find a more precise change position. In such a case, we recommend to simplify the method by skipping the change localization process. As a result, the method detects changes, removes redundant change points (using the clustering technique) and identifies root causes of changes.

8. Conclusion

This paper presented DTMC-CDD, a concept drift detector employing seven components of DTMC to detect, localize and rationalize concept drifts within business processes. The method offers several benefits. It is independent of prior knowledge about the instance labels (i.e., unsupervised). It can identify precise drift positions and rationalize their occurrence. The method was demonstrated to work well under both sudden and gradual drifts, and for changes in both process structure (insertions, deletions, substitutions, loops, terminating states, etc.) and customer behaviour (expectation, preference, etc.) of control-flow. The simulation results demonstrate that the proposed DTMC-CDD method is likely well suited to various business process scenarios.

In its present form, the proposed method can discover changes in a stream of process instances (i.e., offline learning), but lacks mechanisms for dealing with streams of events (i.e., online learning). Therefore, a direction for future work is to extend the approach into an online learning framework. A possible solution is to reorganize this method in an incremental learning form, for example, updating transition probabilities based on new data batches instead of recalculating them. With the goal of detecting, localizing and rationalizing drifts in real-time, we may need to simplify our method, for example, using communication class only to characterize process loop structures. Detecting process drift in other process attributes such as time/duration and resources is also of our interest. A possible solution to detect time duration-related concept drift is to compare the data distribution of two populations, for example, using a Kolmogorov-Smirnov test. Resource changes can be modelled and monitored by applying our DTMC-CDD model as the data is categorical, the same as process activities. Another direction for future research is to enhance the method within scenarios when a label is provided for a process instance (i.e., supervised learning applications). A possible solution is to consider the labels as visible variables and the activities as hidden states to build a hidden Markov model. In this way, we can detect concept drifts in the joint distribution of process instances and their targets.

Acknowledgment

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

References

Adams, J. N., Zelst, S. J. v., Quack, L., Hausmann, K., Van Der Aalst, W. M., & Rose, T. (2021). A framework for explainable concept drift detection in process mining. In *International Conference on Business Process Management* (pp. 400–416). Springer. doi:https://doi.org/10.1007/978-3-030-85469-0_25.

Alippi, C., Ntalampiras, S., & Roveri, M. (2012). An hmm-based change detection method for intelligent embedded sensors. In *The 2012 International Joint Conference on Neural Networks (IJCNN)* (pp. 1–7). IEEE. doi:https://doi.org/10.1109/ijcnn.2012.6252610.

Borges, J., & Levene, M. (2008). Detecting concept drift in web usage mining. In *Proceeding of the Workshop on Web Mining and Web Usage Analysis (WEBKDD)* (pp. 98–110).

Bose, R. J. C., & Van Der Aalst, W. M. (2009). Trace clustering based on conserved patterns: Towards achieving better process models. In *International Conference on Business Process Management* (pp. 170–181). Springer. doi:https://doi.org/10.1007/978-3-642-12186-9_16.

Bose, R. J. C., Van Der Aalst, W. M., Žliobaitė, I., & Pechenizkiy, M. (2013). Dealing with concept drifts in process mining. *IEEE transactions on neural networks and learning systems*, *25*(1), 154–171. doi:https://doi.org/10.1109/tnnls.2013.2278313.

Ceravolo, P., Tavares, G. M., Junior, S. B., & Damiani, E. (2020). Evaluation goals for online process mining: a concept drift perspective. *IEEE Transactions on Services Computing*, (pp. 1–1). doi:https://doi.org/10.1109/TSC.2020.3004532.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, *46*(4), 1–37. doi:https://doi.org/10.1145/2523813.

Hompes, B., Buijs, J. C., Van Der Aalst, W. M., Dixit, P. M., & Buurman, H. (2015). Detecting change in processes using comparative trace clustering. In *5th International Symposium on Data-driven Process Discovery and Analysis* (pp. 95–108). Springer.

Maaradji, A., Dumas, M., La Rosa, M., & Ostovar, A. (2017). Detecting sudden and gradual drifts in business processes from execution traces. *IEEE Transactions on Knowledge and Data Engineering*, *29*(10), 2140–2154. doi:https://doi.org/10.1109/tkde.2017.2720601.

Maisenbacher, M., & Weidlich, M. (2017). Handling concept drift in predictive process monitoring. In *2017 IEEE International Conference on Services Computing (SCC)* (pp. 1–8). IEEE. doi:https://doi.org/10.1109/scc.2017.10.

Mannhardt, F. (2018). *Multi-perspective Process Mining*. Technische Universiteit Eindhoven.

Mannhardt, F., de Leoni, M., Reijers, H. A., & Van Der Aalst, W. M. (2017). Data-driven process discovery-revealing conditional infrequent behavior from event logs. In *International Conference on Advanced Information Systems Engineering* (pp. 545–560). Springer. doi:https://doi.org/10.1007/978-3-319-59536-8_34.

Martjushev, J., Bose, R. J. C., & Van Der Aalst, W. M. (2015). Change point detection and dealing with gradual and multi-order dynamics in process mining. In *International Conference on Business Informatics Research* (pp. 161–178). Springer. doi:https://doi.org/10.1007/978-3-319-21915-8_11.

Meyn, S. P., & Tweedie, R. L. (2012). *Markov chains and stochastic stability*. Springer Science & Business Media.

Norris, J. R. (1998). *Markov chains*. Cambridge university press.

Nuzzo, R. (2014). Statistical errors. *Nature*, *506*, 150–152. doi:https://doi.org/10.1038/506150a.

Ostovar, A., Leemans, S. J., & Rosa, M. L. (2020). Robust drift characterization from event streams of business processes. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *14*(3), 1–57. doi:https://doi.org/10.1145/3375398.

Ostovar, A., Maaradji, A., La Rosa, M., & ter Hofstede, A. H. (2017). Characterizing drift from event streams of business processes. In *International*

Conference on Advanced Information Systems Engineering (pp. 210–228). Springer. doi:`https://doi.org/10.1007/978-3-319-59536-8_14`.

Read, T. R., & Cressie, N. A. (2012). *Goodness-of-fit statistics for discrete multivariate data*. Springer Science & Business Media.

Rojas, E., Munoz-Gama, J., Sepúlveda, M., & Capurro, D. (2016). Process mining in healthcare: A literature review. *Journal of biomedical informatics*, *61*, 224–236. doi:`https://doi.org/10.1016/j.jbi.2016.04.007`.

Roveri, M. (2019). Learning discrete-time markov chains under concept drift. *IEEE transactions on neural networks and learning systems*, *30*(9), 2570–2582. doi:`https://doi.org/10.1109/tnnls.2018.2886956`.

Sartea, R., Farinelli, A., & Murari, M. (2019). Agent behavioral analysis based on absorbing markov chains. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems* (pp. 647–655).

Schubert, E., Sander, J., Ester, M., Kriegel, H. P., & Xu, X. (2017). Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS)*, *42*(3), 1–21. doi:`https://doi.org/10.1145/3068335`.

Seeliger, A., Nolle, T., & Mühlhäuser, M. (2017). Detecting concept drift in processes using graph metrics on process graphs. In *Proceedings of the 9th Conference on Subject-Oriented Business Process Management* (pp. 1–10). doi:`https://doi.org/10.1145/3040565.3040566`.

Serfozo, R. (2009). *Basics of applied stochastic processes*. Springer Science & Business Media.

Sheskin, D. J. (2003). *Handbook of parametric and nonparametric statistical procedures*. Chapman and Hall/CRC.

Spedicato, G. A. (2017). Discrete time markov chains with r. *The R Journal*, *9*(2), 84–104.

Tavares, G. M., Ceravolo, P., Da Costa, V. G. T., Damiani, E., & Junior, S. B. (2019). Overlapping analytic stages in online process mining. In *2019 IEEE International Conference on Services Computing (SCC)* (pp. 167–175). IEEE. doi:`https://doi.org/10.1109/scc.2019.00037`.

Tharwat, A. (2021). Classification assessment methods. *Applied Computing and Informatics*, *17*(1), 168–192. doi:https://doi.org/10.1016/j.aci.2018.08.003.

Velavan, T. P., & Meyer, C. G. (2020). The covid-19 epidemic. *Tropical medicine & international health*, *25*(3), 278–280. doi:https://doi.org/10.1111/tmi.13383.

Xie, A., & Beerel, P. A. (1998). Efficient state classification of finite-state markov chains. *IEEE transactions on computer-aided design of integrated circuits and systems*, *17*(12), 1334–1339. doi:https://doi.org/10.1145/277044.277202.

Yang, L., McClean, S., Donnelly, M., Khan, K., & Burke, K. (2020). Analysing business process anomalies using discrete-time markov chains. In *IEEE 6th International Conference on Data Science and Systems* (pp. 1258–1265). IEEE. doi:https://doi.org/10.1109/hpcc-smartcity-dss50907.2020.00163.

Yeshchenko, A., Di Ciccio, C., Mendling, J., & Polyvyanyy, A. (2019). Comprehensive process drift detection with visual analytics. In *International Conference on Conceptual Modeling* (pp. 119–135). Springer. doi:https://doi.org/10.1007/978-3-030-33223-5_11.

Zheng, C., Wen, L., & Wang, J. (2017). Detecting process concept drifts from event logs. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"* (pp. 524–542). Springer. doi:https://doi.org/10.1007/978-3-319-69462-7_33.

Žliobaitė, I., Pechenizkiy, M., & Gama, J. (2016). An overview of concept drift applications. In *Big data analysis: new algorithms for a new society* (pp. 91–114). Springer. doi:https://doi.org/10.1007/978-3-319-26989-4_4.