Change Point Detection and Dealing with Gradual and Multi-Order Dynamics in Process Mining

J. Martjushev[1,2], R.P. Jagadeesh Chandra Bose[2] and Wil M.P. van der Aalst[2]
martjushev@gmail.com, jcbose@gmail.com, w.m.p.v.d.aalst@tue.nl

[1] University of Tartu, Estonia
[2] Eindhoven University of Technology, The Netherlands.

Abstract. In recent years process mining techniques have matured. Provided that the process is stable and enough example traces have been recorded in the event log, it is possible to discover a high-quality process model that can be used for performance analysis, compliance checking, and prediction. Unfortunately, most processes are not in steady-state and process discovery techniques have problems uncovering "second-order dynamics" (i.e., the process itself changes while being analyzed). This paper describes an approach to discover a variety of *concept drifts* in processes. Unlike earlier approaches, we can discover *gradual drifts* and *multi-order dynamics* (e.g., recurring seasonal effects mixed with the effects of an economic crisis). We use a novel adaptive windowing approach to robustly localize changes (gradual or sudden). Our extensive evaluation (based on objective criteria) shows that the new approach is able to efficiently uncover a broad range of drifts in processes.

1 Introduction

In today's dynamic marketplace, organizations are expected to be flexible and quickly adapt to changing circumstances so as to reduce costs and to improve performance. New legislations such as the WABO act [10] and the Sarbanes-Oxley Act [15], extreme variations in supply and demand, seasonal effects, natural calamities and disasters, deadline escalations [2], etc., force organizations to change their processes. Processes may change suddenly or gradually. The drift may be periodic (e.g., due to seasonal influences) or one-of-a-kind (e.g., the effects of new legislation). For process management it is crucial to discover and understand such *concept drifts* in processes.

Processes executed in today's world are often supported and controlled by information systems, which record events in the form of *event logs*. Process mining aims to *discover*, *monitor* and *improve* real-life processes by extracting knowledge from event logs [1]. Although most business processes change over time, contemporary process mining techniques cannot capture such "second-order dynamics" and analyze these processes as if they are in *steady-state*. However, *detecting and understanding concept drifts is of imminent importance for organizations.*

Although the topic of concept drift is well-studied in various branches of the data mining and machine learning community [3, 12, 16, 21], it has only been recently introduced in the context of process mining [4]. Contemporary concept drift approaches focus on changes in relatively simple structures (e.g., data values and frequencies). Process models are complex artifacts describing behavior involving concurrency, choices, loops, cancelation, etc. Traditional approaches cannot be used to discover the "process of process change". There are three main topics when dealing with concept drifts in process mining [4]:

– *Change (point) detection:* the first and most fundamental problem is to detect that a process change has taken place. If so, the next step is to identify the time periods at which changes have taken place.
– *Change localization and characterization:* once a point of change has been identified, the next step is to characterize the nature of change, and identify the region(s) of change (localization) in a process.
– *Change process discovery:* unraveling the evolution of a process, i.e., the discovery of the change process describing the "second-order" dynamics.

In this paper, we build on the approach in [4] and present novel techniques for detecting process changes and the points at which they changed by analyzing event logs. More specifically, (i) we present an approach to *automatically identify the points (time periods) of change*, (ii) we propose a technique for *change detection using adaptive windows*, (iii) we characterize the notions of *gradual drifts* and *multi-order dynamics* and propose techniques for detecting them, and (iv) we propose an *objective* evaluation framework for change detection. The proposed techniques have been implemented as the Concept Drift plug-in in ProM.[3] The approach was evaluated using a variety of synthetic and real-life event logs.

The remainder of this paper is organized as follows. Section 2 provides the background on change detection techniques based on hypothesis tests. Section 3 presents an approach for automatically identifying change points. An adaptive windowing technique for change detection is presented in Section 4. Section 5 characterizes the notions of gradual and multi-order changes and presents techniques for detecting them. Section 6 proposes an approach for evaluating change detection techniques objectively. Section 7 presents and discusses experimental results. Related work is presented in Section 8. Finally, Section 9 concludes the paper.

2 Background

In this section, we present a brief overview of the change detection technique presented in [4] upon which the concepts presented in this paper are based.

Processes can change with respect to the three main process perspectives, viz., control-flow, data, and resource. Such changes are perceived to induce a drift in the concept (process behavior), e.g., in the way which activities are executed *when*, *how*, and by *whom*. One can consider an event log $\mathcal{L}$ as a time series of traces (traces ordered based on the timestamp of the first event). The basic premise in handling concept drifts is that *the characteristics of the traces before the change point differ from the characteristics of the traces after the change point*. The problem of change (point) detection is then to identify the points in time when the process has changed, if any. Change point detection involves two primary steps: (i) capturing the characteristics of the traces, and (ii) identifying when these characteristics change.

The control-flow perspective of a process characterizes the relationships between activities. Dependencies between activities in an event log can be captured and expressed using the *follows* (or *precedes*) relationship, also referred to as *causal footprints*. Bose et al. [4] proposed four features characterizing the

[3] See `www.processmining.org` for more information and to download ProM.

control-flow dependencies between activities. These features are shown to be effective in detecting process changes. An event log can be transformed into a data set $\mathcal{D}$, which can be considered as a time series (as depicted in Fig. 1), by these features. Change detection is done by considering a series of successive populations[4] of feature values (of some population size w, see Fig. 1) and investigating if there is a significant difference between two successive populations. The premise is that differences are expected to be perceived at change points provided appropriate characteristics of the change are captured as features. The difference between populations is assessed using *statistical hypothesis testing* [18]. Hypothesis tests yield a significance value (the so-called *p-value*), whose range is between 0 and 1, assessing the validity of the null-hypothesis, which typically states that the two populations come from the same distribution. A plot of p-values corresponding to the trace indices captured by populations is inspected to see if significant differences (and thereby process changes) exist. *The p-values are plotted against the indices at the end of the left populations.* Fig. 2 depicts a representative p-value plot. Process changes stand out as *troughs* in the p-value plot. This approach is effective in detecting sudden drifts as shown in [4, 5].

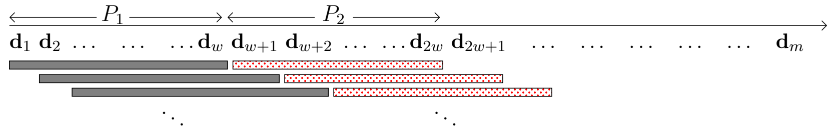

P_1
P_2
$\mathbf{d}_1$ $\mathbf{d}_2$ … … … $\mathbf{d}_w$ $\mathbf{d}_{w+1}$ $\mathbf{d}_{w+2}$ … … $\mathbf{d}_{2w}$ $\mathbf{d}_{2w+1}$ … … … … … $\mathbf{d}_m$

Fig. 1: Basic idea of detecting drifts using hypothesis tests. The data set of feature values is considered as a time series for hypothesis tests. P_1 and P_2 are two populations of size w.

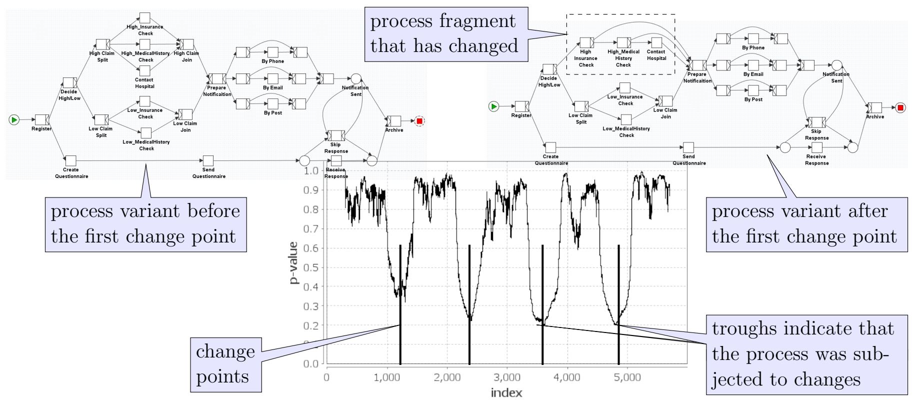


Fig. 2: A plot of p-values of the hypothesis tests. X-axis represents the trace index and Y-axis represents the p-value. Troughs in the plot signify process changes. Process variants before and after a change point can be inspected to identify the fragments that have been changed.

Techniques for dealing with concept drift can be broadly classified into *online* and *offline* depending on whether or not the presence of changes or the occurrence of drifts needs to be uncovered in real-time. In this paper, we focus on offline drift detection. However, our techniques can easily be adapted to the online setting.

[4] A moving window is used to generate the series of populations.

3 Change Point Detection

Process changes manifested in an event log are detected by inspecting the p-value plot of the hypothesis tests over feature values captured for the traces [4].[5] This suffers from the limitation that change and change point detection both need to be done manually by visual inspection. In this section, we present *an automated approach for detecting the points of change.* The basic idea is to first choose an a priori threshold for p-value, $\hat{p}$, to detect the presence of changes. If the significance value (i.e., p-value) of the hypothesis test for two populations P_1 and P_2 is less than $\hat{p}$, we report that there is a change in the process. Having detected that the process has changed, we further explore the two populations to identify the *closest point* of change using a *recursive bisection.* Algorithm 1 sketches the basic idea while Fig. 3 illustrates this. Step 5 of Algorithm 1 facilitates the recursive exploration of the change point search to the closest trace index. Assuming that the p-value is minimum corresponding to the two populations on the right hand side and is less than the threshold $\hat{p}$ (Step 4, Algorithm 1), Fig. 3(b) depicts the recursive search for change point within the right population.

Algorithm 1 Change Point Detection

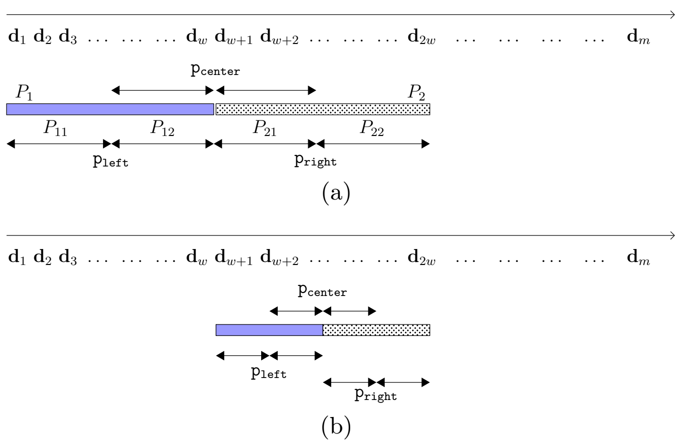
```
1: Let $P_1$ and $P_2$ be the two populations where we have detected a change (i.e., its
   hypothesis test's p-value $< \hat{p}$).
2: Split the two populations $P_1$ and $P_2$ into halves, $P_{11}$ and $P_{12}$ for $P_1$ and $P_{21}$ and
   $P_{22}$ for $P_2$.
3: Apply a hypothesis test on the left ($P_{11}$ and $P_{12}$), center ($P_{12}$ and $P_{21}$), and right
   ($P_{21}$ and $P_{22}$) population pairs as illustrated in Fig. 3(a). Let $p_{left}$, $p_{center}$, and $p_{right}$
   be their respective p-values.
4: Let $p_{\min} = \min\{p_{left}, p_{center}, p_{right}\}$. Let $P^1_{\min}$ and $P^2_{\min}$ be the corresponding popu-
   lations of $p_{\min}$.
5: If $p_{\min} < \hat{p}$, set $P_1 = P^1_{\min}$ and $P_2 = P^2_{\min}$, goto Step 1, else return the index/time
   point corresponding to the trace at end of $P_1$ as the change point.
```

Once a change point has been detected, analysis proceeds as before using a sliding window starting from the first index after the end of the right population. Compared to [4], the location of the change point can be determined more precisely.

4 Adaptive Windowing Approach for Change Detection

The statistical hypothesis test analysis for change detection discussed above uses a *fixed* population size. The goodness of the results depends on the population size, which is largely dependent on the application and the focus of analysis. Typically, one sees a lot of noise in the p-value plot for small populations and the plot tends to be smooth as the population size increases. This can be attributed to the fact that as the population size increases (i.e., as we consider more cases), the variability in the nature of cases reduces and attains a stability. A small population size might result in false positives, i.e., detecting concept drifts that do not exist, while a large population size might result in false negatives, i.e., drifts remain undetected. In order to address this issue, we propose the use of *adaptive windows* where in the population sizes are automatically adapted based

[5] the presence of troughs in the p-value plot indicates that the process was subjected to changes



$\mathbf{d}_1\ \mathbf{d}_2\ \mathbf{d}_3 \dots \dots \dots \mathbf{d}_w\ \mathbf{d}_{w+1}\ \mathbf{d}_{w+2} \dots \dots \dots \mathbf{d}_{2w} \dots \dots \dots \dots \mathbf{d}_m$
p_{center}
P_1
P_2
P_{11}
P_{12}
P_{21}
P_{22}
p_{left}
p_{right}
(a)
(b)

Fig. 3: Basic idea of our recursive bisection approach to detect and localize change points. (a) change point search by considering the left, center, and right sub-populations (b)recursive search for change point in the right population.

on the characteristics of the data stream. We adapt the ADWIN technique [3], an approach for online change detection using an adaptive size sliding detection window. Algorithm 2 presents the adaptive window approach for change detection. The basic idea is to use minimum and maximum size limits for populations and extend the population sizes until a change has been detected or the populations reach the maximum size limit. If the maximum size is reached, we discard the historically old data (i.e., the left most population) and proceed with the hypothesis tests with recent data. When a change has been detected, we identify the change point and proceed with hypothesis tests using two new smaller populations (of minimum size) with the new left population starting at the first index after the old right population and the new right population starting at the first index after the new left population (Steps 5-6, Algorithm 2).

As mentioned earlier, the p-values are plotted against the indices at the end of the left populations. When a change has been detected or when the populations reach a maximum size, we create new populations. This creates a gap between the indices at the end of the old and new left populations. To have a continuous p-value plot, we connect the p-values at the old and new indices (corresponding to the old and new left populations) by a straight line.

5 Dealing with Gradual Drifts and Multi-Order Dynamics

In this section, we characterize the notions of *gradual* and *multi-order changes* and present techniques for detecting such changes.

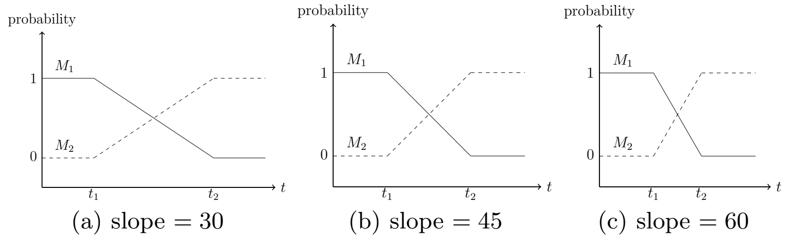
5.1 Gradual Drifts

In gradual drifts, one concept fades gradually while the other takes over. This phenomenon of gradual change can be modeled in many ways by means of functions that describe how things grow or decay as time passes. For example, the change can be linear between two sources as illustrated in Fig. 4. In the figure, initially until t_1, only the process variant M_1 is in operation, i.e., all cases em-

Algorithm 2 Change Detection Using Adaptive Windows

```
Require: a minimum population size $w_{\min}$, a maximum population size $w_{\max}$, p-value
    threshold $\hat{p}$, a step size $k$, and a data stream of values $\mathcal{D}$
 1: Let $P_{\texttt{left}}$ and $P_{\texttt{right}}$ be two populations of size $w_{\min}$ with $P_{\texttt{right}}$ starting at the first
    index after the end of $P_{\texttt{left}}$.
 2: repeat
 3:     Apply a hypothesis test over $P_{\texttt{left}}$ and $P_{\texttt{right}}$. Let $p$ be its p-value.
 4:     if $p < \hat{p}$ then
 5:         Identify the change point within $P_{\texttt{left}}$ and $P_{\texttt{right}}$ using Algorithm 1.
 6:         Create two new populations $P'_{\texttt{left}}$ and $P'_{\texttt{right}}$ of size $w_{\min}$ with $P'_{\texttt{left}}$ starting
    at the first index after the end of $P_{\texttt{right}}$ and $P'_{\texttt{right}}$ starting at the first index after
    the end of $P'_{\texttt{left}}$. Set $P_{\texttt{left}} = P'_{\texttt{left}}$ and $P_{\texttt{right}} = P'_{\texttt{right}}$.
 7:     else
 8:         Extend the left and right populations by a step size $k$. Reassign the right
    population to start at the first index after the end of the extended left population
    $P_{\texttt{left}}$.
 9:         if the size of the population $\geq w_{\max}$ then discard the left population $P_{\texttt{left}}$.
    Split the right population $P_{\texttt{right}}$ into two halves and use them as the left and right
    populations.
10:         end if
11:     end if
12: until the end of $P_{\texttt{right}}$ doesn't reach the end of $\mathcal{D}$
```

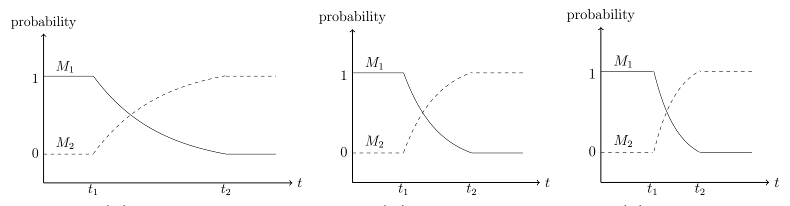
anate from M_1 until t_1. The gradual change between process variants M_1 and M_2 happen *linearly* between t_1 and t_2, i.e., the cases from M_1 and M_2 constantly decrease and increase respectively. The degree of decrease/increase is characterized by the *slope* as illustrated in Fig. 4(a)–(c). Subsequently, from t_2, all cases emanate only from M_2. As another example, one can notice an *exponential rate of increase/decrease* of cases from two processes as illustrated in Fig. 5. Here, the degree of change between t_1 and t_2 is characterized by the function $e^{-\lambda t}$ for M_1. The probability of a case emanating from M_2 at any instant of time between t_1 and t_2 is modeled as $1 - p_{M_1}$ where p_{M_1} is the probability of a case being from M_1.



(a) slope = 30 (b) slope = 45 (c) slope = 60

Fig. 4: Different variants of linear gradual drift between t_1 and t_2 for processes M_1 and M_2. The rate of change is characterized by the slope.

Analyzing event logs for concept drifts under gradual changes can be done using hypothesis tests on features characterizing the process execution behavior. However, since the transition between two processes M_1 and M_2 is gradual/smooth, the p-values tend to be higher compared to sudden drifts if the conventional sliding window approach is considered. Furthermore, if detected,



(a) $\lambda = 0.5$ (b) $\lambda = 1.0$ (c) $\lambda = 1.5$

Fig. 5: Different variants of exponential gradual drift between t_1 and t_2 characterized by the function $e^{-\lambda t}$ for M_1.

the troughs tend to be wider, in proportion to the duration for which the gradual change is operational. Therefore, to detect gradual drifts, we advocate the use of *non-continuous populations* as illustrated in Fig. 6. The intuition behind this is that the *gap* between populations makes it easier to pick up differences even when instances from both processes co-exist. Provided a proper choice of the gap is made, at the onset of gradual change, the populations P_1 and P_2 capture instances only from M_1 and M_2 respectively; a hypothesis test on these two populations should yield a significantly lower p-value, thus facilitating the detection of a change.

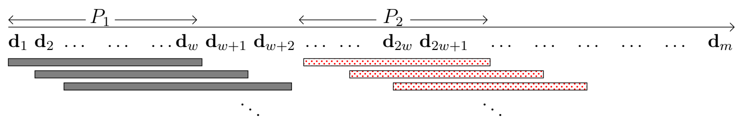

P_1
P_2
$\mathbf{d}_1$ $\mathbf{d}_2$ … … … $\mathbf{d}_w$ $\mathbf{d}_{w+1}$ $\mathbf{d}_{w+2}$ … … $\mathbf{d}_{2w}$ $\mathbf{d}_{2w+1}$ … … … … … $\mathbf{d}_m$

Fig. 6: General idea of gradual drift detection using hypothesis tests on non-continuous populations: P_1 and P_2 are deliberately separated by a gap.

5.2 Multi-Order Dynamics

In this section, we extend the notion of process changes to also include *multi-order dynamics* where process changes can happen at multiple levels of (time) granularity, e.g., weekly and yearly recurring drifts may be mixed with drifts due to economic developments. For example, suppose an organization induces a process change from M_1 to M_2 after 24 weeks. Furthermore, let us assume that there are two variants of process M_1, viz., M_{11} and M_{12}, and two variants of process M_2, viz., M_{21} and M_{22}, which the organization recurringly changes every six weeks within/after the first 24 weeks respectively.[6] Fig. 7 illustrates this phenomenon. Each time unit in the figure corresponds to six weeks. We can see that process changes happen at two-levels in this example. One change induced every six weeks and another change happening at 24 weeks (time unit 4 in the figure) between M_1 and M_2. When dealing with concept drifts, one has to take into consideration the presence of such multi-order dynamics. The framework for change detection using hypothesis tests can still be used for detecting multi-order changes. However, *instead of considering populations based on a fixed volume of traces/cases, we should consider populations at different time scales.* Populations

[6] To simplify discussion, the duration for which a process variant is active is kept uniform. However, in reality, processes can be deployed for varying durations and can be changed at varying intervals.

comprising cases within shorter-time periods are to be used for detecting *micro-level* changes while larger time periods are to be chosen for *macro-level* changes. For example, using populations comprising of cases in a time period up to six weeks, we would be able to detect the seven *micro-level* change points in Fig. 7. Instead, if we choose the populations to comprise of cases from a time period between 12 weeks and 24 weeks, we would be able to detect the single *macro-level* change between M_1 and M_2 at time unit 4 in Fig. 7.

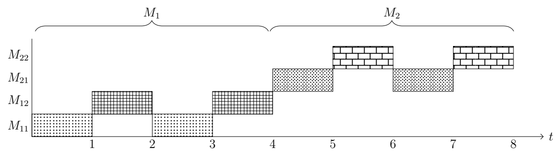


Fig. 7: Illustration of multi-order changes involving four process variants: micro-level drifts (e.g., the alternation of M_{11} and M_{12}) operate at a different time scale than the macro-level drift from M_1 to M_2.

6 Objective Evaluation of Change Detection Techniques

The automatic detection of change points proposed in Section 3 provides an *objective evaluation mechanism* for change detection techniques. We adopt classic metrics in data mining such as the number of true positives (TP), false positives (FP) and false negatives (FN), and derived metrics from these, viz., precision, recall, and $F1$-score as objective measures. In order to define these metrics, we use a lag period l surrounding a detected or actual change point. The interpretation of these metrics (see Fig. 8) is as follows:

- *TP:* a change point is detected at $\hat{t}$ and there is an actual change within $\hat{t} \pm l$
- *FP:* a change point is detected at $\hat{t}$ but there is no actual change within $\hat{t} \pm l$
- *FN:* an actual change happened at $\hat{t}$ but no change has been detected within $\hat{t} \pm l$

Precision measures the fraction of detected changes that are correct while *recall* measures the fraction of actual changes that have been detected. In other words, *precision* $= TP/(TP + FP)$ and *recall* $= TP/(TP + FN)$. A measure that combines both precision and recall is the *F1-score*, defined as the harmonic mean between precision and recall, i.e., *F1-score* $= 2.$*precision.recall*$/($*precision*$+$ *recall*$)$. *Techniques that are able to detect changes with a high precision and recall (both close to 1.0) are preferred over others.*

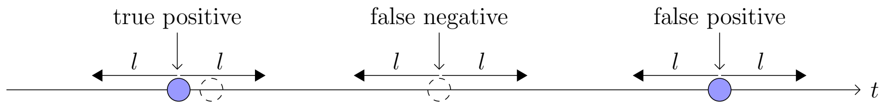


Fig. 8: Objective evaluation of change detection techniques. The solid circle and the dashed circle indicate a detected change and an actual change respectively.

7 Experiments and Discussion

The various concepts presented in this paper have been implemented in the Concept Drift plug-in in ProM. In this section, we discuss the results of applying these concepts for change (point) detection on several logs.

7.1 Sudden Drifts

We use the synthetic insurance claim event log [4] to assess the goodness of the change and change point detection techniques proposed in this paper. This event log contains 6000 traces and 58783 events distributed over 15 activities and incorporates a sudden drift phenomenon over five process variants. The event log contains 1200 traces from each process variant with the change points induced at 1200, 2400, 3600, and 4800. The approach presented in [4] was shown to detect the presence of all the four changes. However, the change points had to be detected manually. We have applied the adaptive windowing approach using the Kolmogorov-Smirnov test (KS-test) [18] over the data stream obtained on the J-measure feature [4] for each activity pair. Fig. 9 depicts the average p-value plot of the KS-test on all activity pairs using a minimum population size of 100, maximum population size of 500, step size of 20 and a p-value threshold of 0.4. The red dots in the plot indicate the *automatically* detected change points using the approach presented in Section 3. The change points are detected at indices 1207, 2415, 3598, and 4793. Using a lag window of 20 traces, we have the following metrics (TP=4, FP=0, FN=0, precision=1.0, and recall=1.0), i.e., we are able to detect all changes within 20 traces of the actual change points, which is quite promising.

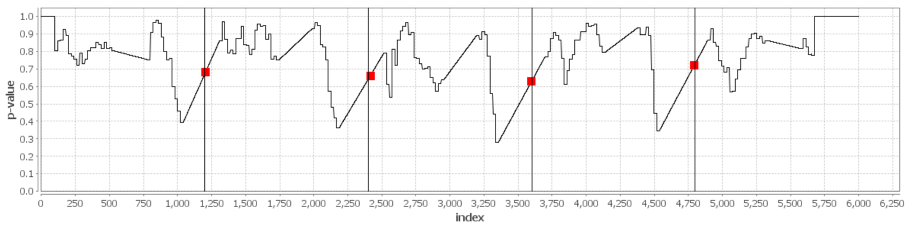
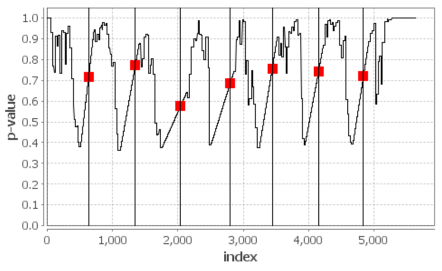


Fig. 9: Average p-value (over all activity pairs) using the adaptive windowing technique for KS-test on the J-measure estimated for each trace. The red dots indicate the *automatically* detected change points. The X-axis represents the trace index and Y-axis represents the p-value of the test. The solid vertical lines indicate the actual change points.

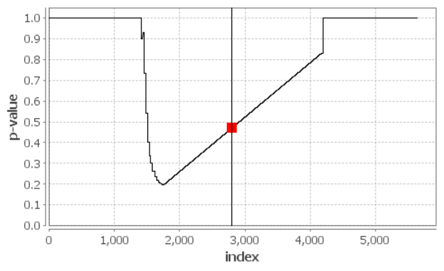
7.2 Multi-Order Dynamics

In order to conduct a controlled experiment involving multi-order changes, we have modeled a process exhibiting drifts at different time scales using CPN tools [8]. We considered four process variants pertaining to the insurance claim example and generated an event log exhibiting multi-order changes as illustrated in Fig. 7. Two of the process model variants recurred alternatively every 6 weeks within the first 24 weeks while the other two process variants recurred alternatively every 6 weeks in the next 24 weeks. In practice, the arrival rate of cases can be different at different time periods. However, for simplicity and ease of discussion, we have modeled instances to arrive at a constant rate over the entire period. Furthermore, instances were modeled to arrive only during working hours and on week days at the rate of approximately 3 instances per hour. The event log contains 5647 cases and 57530 events distributed over 15 activities. There are 7 micro-level drift points induced at indices 629, 1346, 2038, 2802, 3444, 4156, and 4845 in this event log and one macro-level drift at index 2802.

As mentioned earlier, one needs to look at populations in terms of varying time periods rather than the number of traces when dealing with multi-order changes. Micro-level and macro-level drifts can be detected by considering populations comprising of cases defined over shorter and longer time periods respectively. Since the micro-level drifts are induced every 6 weeks, to detect these changes, we need to consider populations that do not exceed 6 weeks. Fig. 10(a) depicts the average p-value of the KS-test on the J-measure over all activity pairs using a minimum population size of 3 days, maximum population size of 6 weeks, step size of 1 day, and a p-value threshold of 0.4. We can see that we are able to detect all the 7 drifts. Also, we are able to automatically detect the exact change points. Fig. 10(b) depicts the average p-value of the KS-test on the J-measure over all activity pairs using a minimum population size of 12 weeks, maximum population size of 24 weeks, step size of 3 days and a p-value threshold of 0.2. We can see that by choosing a longer time period, we are able to detect the lone macro-level drift. The drift point is automatically detected at 2802 (the actual drift point is also 2802), which is accurate.



(a) micro-level drifts



(b) macro-level drift

Fig. 10: Detection of multi-order changes. Average p-value (over all activity pairs) using the adaptive windowing technique for KS-test on the J-measure estimated for each trace. The red dots indicate the automatically detected change points. Both micro-level drifts (left) and macro-level drifts (right) are detected using appropriate time scales. The solid vertical lines indicate the actual change points.

It is imperative to note that the adaptive windowing approach and the change point detection technique are both sensitive to the p-value threshold. Fig. 11 depicts the objective metrics for the micro-level drift detection using a lag period of 2 days (50 traces) for different p-value thresholds. We can see that a choice of low p-value thresholds results in high false negatives (low recall) while a choice of high p-value thresholds results in high false positives (low precision). The $F1$-score increases with increasing p-value thresholds (due to increasing true positives) up to a certain point and deteriorates with further increase of p-value threshold (due to increasing false positives). The absolute p-values of the hypothesis tests are in turn dependent on the degree of change. We notice high p-values in scenarios where the process changes are minimal and concentrated over (affecting) only a few activities in the process. In such cases, the p-values will be less only for those features that involve the activities affected by the change. The high p-values overall is due to the effect of aggregation (average) that we do over all activity pairs. In other words, the absolute p-values are inversely proportional to the extent of change and the number of activities affected by the change.

7.3 Gradual Drifts

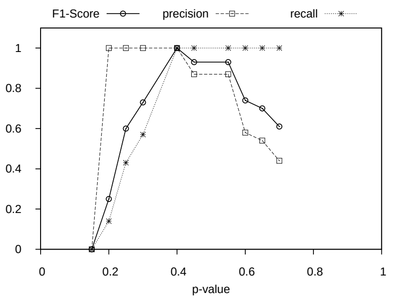
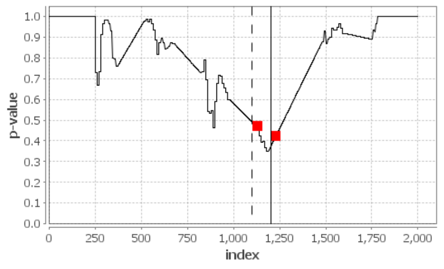
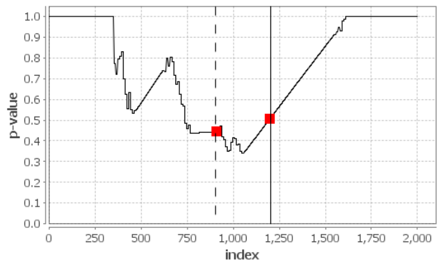


Fig. 11: Influence of p-value threshold on change detection.

We now assess the goodness of the proposed approach in handling gradual drifts. We consider both the scenarios where the change is linear as well as exponential between two sources. Again CPN tools is used to create drifting processes allowing for a controlled experiment. We have used two of the process variants of the insurance claim example [4] and generated event logs with linear and exponential graduality. We first discuss the results on linear graduality. We considered an event log containing 2000 traces and 19346 events distributed over 15 activities. The linear drift was induced between the traces 1100 and 1200. Fig. 12(a) depicts the average p-value obtained using adaptive windowing technique for the KS-test on the J-measure over all activity pairs using a minimum population size of 200, maximum population size of 300, step size of 10, gap size of 100, and a p-value threshold of 0.35. The drifts are detected at 1128 and 1229 (the actual drifts are at 1100 and 1200). Using a gap size of 50 (with the rest of the configuration remaining the same), the drifts are detected at 1158 and 1199. We can also see that for gradual changes, the width of the troughs are wider. We now consider an example of exponential gradual drift. We considered an event log containing 2000 traces and 19183 events distributed over 15 activities. An exponential drift was introduced between 900 and 1200 traces with a decay rate of $\lambda = 0.005$ for the first process variant. Fig. 12(b) depicts the average p-value obtained using adaptive windowing technique for the KS-test on the J-measure over all activity pairs using a minimum population size of 200, maximum population size of 300, step size of 10, gap size of 300, and a p-value threshold of 0.35. The drifts are detected at 907 and 1198 (the actual drifts are at 900 and 1200). Using a gap size of 100 (with the rest of the configuration remaining the same), the drifts are detected at 907 and 998. By choosing an appropriate gap size, we are able to detect the change points very close to the actual change points.



(a) linear gradual drift



(b) exponential gradual drift

Fig. 12: Detection of gradual changes using non-continuous populations. The red dots indicate the detected start/end of gradual change. The dashed and solid vertical lines indicate the start and end respectively of the actual gradual change.

7.4 Case Study

We have applied the concepts presented in this paper on several real-life event logs. In this section, we discuss the results of one such experiment of analyzing drifts from the all-in-one-permit handling process of a large Dutch municipality. Since October 1, 2010, the All-in-one Permit for Physical Aspects (omgevingsvergunning) has come into force through the WABO act [10]. This entails an overarching procedure for granting permission for projects like the construction, alteration or use of a house or building, etc. Now, the municipalities have one permit, one procedure and one set of submittal requirements, followed by one legal remedies procedure and enforcement by one authority. We considered an event log containing 184 cases and 4391 events distributed over 38 event classes (activities). The cases arrived between 19 Oct 2010 and 29 Dec 2011. We have applied the adaptive windowing approach with minimum and maximum population sizes of 10 and 30 respectively and a p-value threshold of 0.3 on the J-measure feature over all activity pairs. Fig. 13 depicts the resulting drift plot along with the drift points. We see that there are three change points pertaining to traces at indices 50, 113, and 157. We have partitioned the event log into four parts, $\mathcal{L}_0$ comprising of cases from 1 to 49, $\mathcal{L}_1$ comprising of cases from 50 to 112, $\mathcal{L}_2$ comprising of cases from 113 to 156, and $\mathcal{L}_3$ comprising of cases from 157 to 184. We mined process models from these four event logs and analyzed for the changes that transpired between them.

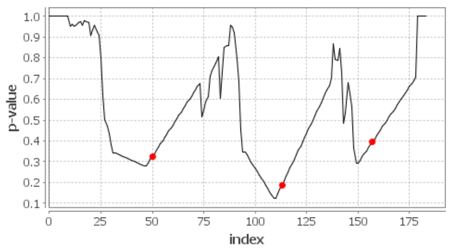
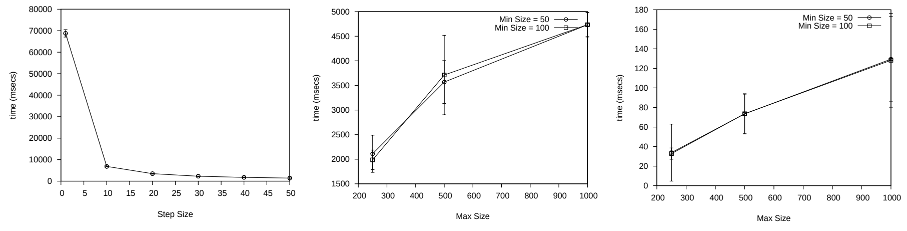


Fig. 13: Average p-value (over all activity pairs) using the adaptive windowing technique for KS-test on the J-measure estimated for each trace. The red dots indicate the detected change points.

The basic process comprises of four high-level steps: (i) registration and acknowledgements, (ii) procedural check 1, (iii) procedural check 2, and (iv) final assessment and decision. The changes primarily correspond to the procedural check steps. There is a change w.r.t two activities viz., suspend the time limit and Procedure change 2 || Activities regular procedure 2 between the two model variants corresponding to logs $\mathcal{L}_0$ and $\mathcal{L}_1$. In the variant mined using $\mathcal{L}_0$, these activities are executed in only 6 and 7 cases respectively out of the 49 cases whereas in the latter these are executed for each and every case. In this variant corresponding to log $\mathcal{L}_2$, the organization introduced some new activities pertaining to shipments and licence exemptions. Furthermore, the activities Waw permit aspect 1 and Waw permit aspect 2 are being phased out. This is noticed in the fact that these activities are executed in only 14 of the 43 cases whereas in its previous variant these activities are executed for each and every case. The process variant corresponding to $\mathcal{L}_3$ differs from $\mathcal{L}_2$ in the fact that these two activities are completely phased out.

7.5 Time Complexity Analysis

The change detection technique using hypothesis tests on a data stream of feature values proposed in [4] uses a sliding window that moves by one unit to generate successive populations. The number of hypothesis tests to be performed overall is directly proportional to the number of cases in an event log. Furthermore, this has to be repeated for different data streams, one for each feature (e.g., the J-measure on the follows relation over all activity pairs). For logs with many cases and/or large number of activities, this tends to be computationally expensive. One can improve this by progressing the sliding window by k units rather than one (for some $k \geq 2 \in \mathbb{N}$).[7] Fig. 14(a) depicts the average computational time along with the 95% confidence intervals over five independent runs for the insurance claim log (considering the J-measure over all activity pairs) for a fixed population size and different step sizes. We can see that time complexity reduces k-fold for a step size of k. This novel idea allows us to speed up concept drift analysis significantly with only a minor loss in terms of accuracy.



(a) (b) (c)

Fig. 14: The influence of step size and population sizes on the computational time for change detection and change point search. (a) Influence of step size (b) Influence of min/max size for change detection (ADWIN) (c)Influence of min/max size for drift point search (ADWIN)

The computational complexity of the adaptive windowing technique depends on the min/max population size thresholds since the time for each hypothesis test is dependent on the size of the population.Fig. 14(b) and Fig. 14(c) depict the average computational time along with the 95% confidence intervals over five independent runs for the insurance claim log (considering the J-measure over all activity pairs) on a fixed step size. We can see that the complexity only depends on the maximum population size: doubling the minimum population size has hardly any effect whereas the change detection and localization times depend linearly on the maximum population size.

8 Related Work

Process flexibility has been one of the “hotspots” in BPM/WFM research during the last two decades, e.g., collections of typical change patterns [11, 19], extensive taxonomies of various flexibility approaches and mechanisms [14, 17], and classifications of process changes [13] have been provided. Despite the many publications on flexibility, most process mining techniques assume a process to be in steady state. A notable exception is the approach by Günther et al. [7], which attempts at using process mining to provide an aggregated overview of all changes that have happened so far. However, this approach assumes that change logs are available, i.e., modifications of the workflow model are recorded. At this point in time very few information systems provide such change logs.

This paper builds on [4] where the concept drift problem in process mining was analyzed for the first time. Since [4], several other techniques have been de-

[7] This is incorporated in the adaptive windowing technique (the step size parameter) presented in this paper.

veloped for dealing with concept drifts in process mining [6,9,20]. Carmona and Gavaldà [6] have proposed an online technique for detecting process changes. They first create an abstract representation of the process in the form of polyhedra using the prefixes of some initial traces in the event log. Subsequent traces are sampled and assessed whether they lie within the polyhedra or not. If a sample lies within the polyhedra, it is considered to be from the same process. If significant number of samples lie outside the polyhedra, a process change is said to be detected. Although Carmona and Gavaldà [6] use the adaptive windowing technique like us, their technique differs from our approach in several ways: (i) their approach constructs an abstract representation of a process unlike ours where we consider features characterizing the traces, (ii) their approach is applicable only for change detection whereas our framework is applicable for both change (point) detection and change localization (for change localization, refer to [5]), and (iii) their approach handles only sudden drifts whereas we present techniques for dealing with sudden, gradual, and multi-order dynamics. Furthermore, the tool support provided by the authors does not detect change points and does not work on logs with multiple process changes, i.e., it doesn't detect the presence/absence of multiple changes and doesn't report when (the trace index) process changes have happened. The tool just reports that a change exists and terminates (if changes exist) and does not terminate if no changes exist. In contrast, our plug-in can handle multiple process changes and can detect both the presence of and the points of change in addition to being able to assist in change localization.

Weber et al. [20] attempt at detecting concept changes by comparing models mined from event logs using a sliding window with a representative ground truth model. They use a probabilistic deterministic finite automata (PDFA) as a representation for a process and use statistical tests for detecting if the mined distribution, or its PDFA representation, has changed significantly from the ground truth. One of the challenges with this approach is the number of samples (traces) required to mine a good representative model that can be compared with the ground truth model. In contrast, our approach relies on characteristic differences in the features defined over traces for change detection.

To our best knowledge, we are the first to address the notion of *gradual* and *multi-order* changes in process mining.

9 Conclusions and Future Work

Although most business processes change over time, contemporary process mining techniques tend to analyze these processes as if they are in steady-state. For process management it is crucial to discover and understand such concept drifts in processes. In this paper, we proposed an adaptive windowing technique for change detection and a novel means of detecting change points automatically. Furthermore, we characterized the notions of gradual and multi-order changes and proposed techniques for detecting such changes. Our initial results show that the proposed techniques are very promising, i.e., we are able to detect changes accurately. In this paper, we have considered process changes only from a control-flow perspective. In the future, we would like to extend this to also include data/resource perspective changes. Furthermore, we would like to evaluate our approach using additional real-life case studies where concept drift is analyzed at runtime thereby providing users immediate diagnostics regarding recent changes.

Acknowledgments J. Martjushev is grateful to Archimedes Foundation and the Ministry of Education and Research for funding his research through the national scholarship program, Kristjan Jaak.

References

1. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer New York Inc (2011)
2. van der Aalst, W.M.P., Rosemann, M., Dumas, M.: Deadline-based Escalation in Process-Aware Information Systems. Decision Support Systems 43(2), 492–511 (2011)
3. Bifet, A., Gavaldà, R.: Learning from Time-Changing Data with Adaptive Windowing. In: Proceedings of the SIAM Data Mining Conference. pp. 443–448 (2007)
4. Bose, R.P.J.C., van der Aalst, W.M.P., Žliobaitė, I., Pechenizkiy, M.: Handling Concept Drift in Process Mining. In: CAiSE. LNCS, vol. 6741, pp. 391–405. Springer, Berlin (2011)
5. Bose, R.P.J.C.: Process Mining in the Large: Preprocessing, Discovery, and Diagnostics. Ph.D. thesis, Eindhoven University of Technology (2012)
6. Carmona, J., Gavaldà, R.: Online Techniques for Dealing with Concept Drift in Process Mining. In: IDA. LNCS, vol. 7619, pp. 90–102 (2012)
7. Günther, C.W., Rinderle-Ma, S., Reichert, M., van der Aalst, W.M.P.: Using Process Mining to Learn from Process Changes in Evolutionary Systems. International Journal of Business Process Integration and Management 3(1), 61–78 (2008)
8. Jensen, K., Kristensen, L.: Coloured Petri Nets: Modelling and Validation of Concurrent Systems. Springer (2009)
9. Luengo, D., Sepúlveda, M.: Applying Clustering in Process Mining to Find Different Versions of a Business Process That Changes over Time. In: BPM Workshops (1). LNBIP, vol. 99, pp. 153–158 (2012)
10. Ministerie van Infrastructuur en Milieu: All-in-one Permit for Physical Aspects: (Omgevingsvergunning) in a Nutshell (2010)
11. Mulyar, N.: Patterns for Process-Aware Information Systems: An Approach Based on Colored Petri Nets. Ph.D. thesis, Eindhoven University of Technology (2009)
12. Pechenizkiy, M., Bakker, J., Žliobaitė, I., Ivannikov, A., Kärkkäinen, T.: Online Mass Flow Prediction in CFB Boilers with Explicit Detection of Sudden Concept Drift. SIGKDD Explorations 11(2), 109–116 (2009)
13. Ploesser, K., Recker, J.C., Rosemann, M.: Towards a Classification and Lifecycle of Business Process Change. In: BPMDS. vol. 8 (2008)
14. Regev, G., Soffer, P., Schmidt, R.: Taxonomy of Flexibility in Business Processes. In: Business Process Modeling, Development, and Support (2006)
15. Sarbanes, P., G. Oxley et. al.: Sarbanes-Oxley Act of 2002 (2002)
16. Schlimmer, J., Granger, R.: Beyond Incremental Processing: Tracking Concept Drift. In: Proceedings of the Fifth National Conference on Artificial Intelligence. vol. 1, pp. 502–507 (1986)
17. Schonenberg, H., Mans, R., Russell, N., Mulyar, N., van der Aalst, W.M.P.: Process Flexibility: A Survey of Contemporary Approaches. In: Advances in Enterprise Engineering I. LNBIP, vol. 10, pp. 16–30. Springer (2008)
18. Sheskin, D.: Handbook of Parametric and Nonparametric Statistical Procedures. Chapman & Hall/CRC (2004)
19. Weber, B., Rinderle, S., Reichert, M.: Change Patterns and Change Support Features in Process-Aware Information Systems. In: CAiSE. LNCS, vol. 4495, pp. 574–588. Springer (2007)
20. Weber, P., Bordbar, B., Tino, P.: Real-Time Detection of Process Change using Process Mining. In: Imperial College Computing Student Workshop. Department of Computing Technical Report, vol. DTR11-9, pp. 108–114 (2011)
21. Widmer, G., Kubat, M.: Learning in the Presence of Concept Drift and Hidden Contexts. Machine Learning 23(1), 69–101 (1996)