Process Variant Analysis Across Continuous Features: A Novel Framework⋆

Ali Norouzifar[1][0000−0002−1929−9992], Majid Rafiei[1][0000−0001−7161−6927], Marcus Dees[2][0000−0002−6555−320X], and Wil van der Aalst[1][0000−0002−0955−6940]

[1] RWTH University, Aachen, Germany
{ali.norouzifar, majid.rafiei, wvdaalst}@pads.rwth-aachen.de
[2] UWV Employee Insurance Agency, Amsterdam, Netherlands
Marcus.Dees@uwv.nl

Abstract. Extracted event data from information systems often contain a variety of process executions making the data complex and difficult to comprehend. Unlike current research which only identifies the variability over time, we focus on other dimensions that may play a role in the performance of the process. This research addresses the challenge of effectively segmenting cases within operational processes based on continuous features, such as duration of cases, and evaluated risk score of cases, which are often overlooked in traditional process analysis. We present a novel approach employing a sliding window technique combined with the earth mover's distance to detect changes in control flow behavior over continuous dimensions. This approach enables case segmentation, hierarchical merging of similar segments, and pairwise comparison of them, providing a comprehensive perspective on process behavior. We validate our methodology through a real-life case study in collaboration with UWV, the Dutch employee insurance agency, demonstrating its practical applicability. This research contributes to the field by aiding organizations in improving process efficiency, pinpointing abnormal behaviors, and providing valuable inputs for process comparison, and outcome prediction.

**Keywords:** process mining · process comparison · business process improvement.

1 Introduction

Process mining techniques are used to analyze event data generated by different types of information systems. For instance, performance analysis using process mining techniques has provided a range of new opportunities for business owners to analyze and improve their processes. From the performance point of view, we often observe that the execution policies may vary significantly for different groups of process instances based on their characteristics.
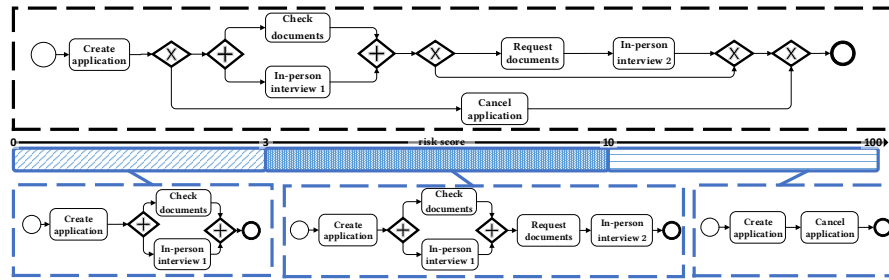
(a) The BPMN on top shows a claim handling process in which each case has a risk score and may have a different handling procedure based on the risk score.
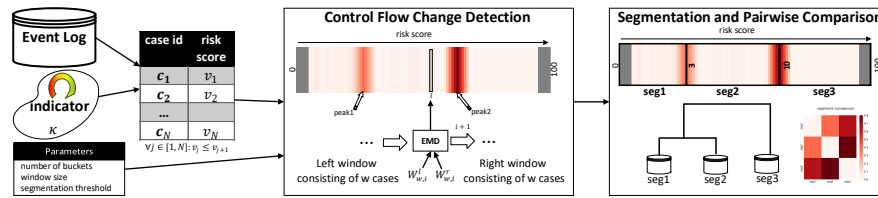
(b) The proposed framework for process variant analysis based on the risk score.

Fig. 1: A claim handling process as a motivating example.

For example, in the claim handling process shown in Fig. 1a, a risk score assigned to the cases affects the handling procedure. On top, the business process explaining the whole process is shown without giving any clue that the process could be different for cases in different ranges of the risk score. After creating an application, cases with a risk score higher than 10 are canceled. The remaining cases go through the check documents step and require an in-person interview, then there is an exclusive choice that decides whether the cases are required to submit more documents and have another interview, or they can skip these two steps if the risk score is lower than 3.

Using the available data in information systems and process analysis tools, different dimensions for each case in the event log can be extracted. The introduced approach in Fig. 1b considers risk score as an example of a continuous dimension and has two main parts. First, the *control flow change detection* is designed by using a sliding window shifting over the dimension range. Then, local peaks are used for *segmentation and pairwise comparison*.

In this paper, we have the following goals: 1) Motivating the problem of finding process variants over the range of a continuous dimension, e.g., risk score or case duration. 2) Introducing a sliding window approach using the earth mover's distance to find the changes in control flow. 3) Case segmentation based on continuous dimensions and control flow, and pairwise comparison of the segments. 4) Testing the usability of the framework with a real-life case study.

The output of the framework is an automated segmentation of cases based on key dimensions influencing the control-flow behavior which can be used as input for other types of analysis, such as process comparison [13], outcome pre-

diction [14], or labeling the traces as desirable or undesirable in order to use them in process discovery algorithms working with desirable and undesirable traces [11,5].

2   Related Work

Extracted event logs from information systems often consist of a variety of process executions containing deviating behaviors, manual interventions, infrequent patterns, process drift over time, and many other inconsistencies resulting in high complexity [6]. While filtering out infrequent behavior can mitigate data complexity [4], it is important to note that infrequent behavior may sometimes include significant deviations that need further investigation. In [8], the earth mover's distance is used to identify distinct cohorts of traces based on trace attributes, offering a framework that explores all trace attributes to uncover combinations leading to the most diverse control-flow behavior. Unlike this paper, we do not need the assumption that the dimensions are discretized in advance.

Trace clustering can also be considered as related work to our research. Different trace clustering approaches may use a control-flow perspective, other available attributes, or a combination of both [7]. In [1], different similarity measures to cluster the traces based on the control flow are introduced and compared. In [17], an event log is divided into sub-event logs considering an attribute value, then hierarchical clustering is used to merge similar clusters. However, choosing a distinctive attribute is not straightforward. In addition, when considering continuous attributes it is even more challenging to set distinctive thresholds. In [15], the active trace clustering framework is introduced which takes process models explaining the clusters into account while doing trace clustering. The trace clustering methods cannot directly provide solutions for our problem, i.e., considering the performance perspective.

Considering time as an important dimension in event data analysis, the identification of concept drift is an interesting research question [12]. These methods usually use a feature space to characterize the control flow and use some techniques to pinpoint the changes. If a change is not observable using the selected feature space, these algorithms fail to detect it. In [16], a concept drift detection algorithm is introduced which leverages declarative constraints to represent control-flow behavior. A multi-variate change point detection algorithm is implemented to find the changes over time. In [10], another method is proposed which is based on the statistical tests applied to the distribution of partially ordered runs in two consecutive time windows. In [2], the earth mover's distance is used to find drifts in the control flow. The mentioned approaches only focus on changes in control flow behavior over time and not on different continuous dimensions. Similar to the framework proposed in [2], we employ a window-based approach using the earth mover's distance function to identify the changes across continuous features. Our framework works effectively without a large feature space, which might struggle to capture all potential changes in control flow. Additionally, it does not use statistical tests that often rely on some assumptions.

3   Preliminaries

$\mathcal{B}(A)$ is the set of all multisets over some set A. Considering $B \in \mathcal{B}(A)$, $B(a)$ denotes the frequency of element $a \in A$. We write $x \in B$ if $B(x) > 0$. The event log is an important concept in our work, therefore we formally introduce it.

Definition 1 (Event Log). *Let $\mathcal{C}$, $\mathcal{A}$, and $\mathcal{T}$ be the universe of case identifiers, the universe of activities, and the universe of timestamps respectively. $e=(c,a,t)$ is a tuple representing an event where $\pi_{\mathcal{C}}(e)=c \in \mathcal{C}$, $\pi_{\mathcal{A}}(e)=a \in \mathcal{A}$, and $\pi_{\mathcal{T}}(e)=t \in \mathcal{T}$. $\mathcal{E}=\mathcal{C} \times \mathcal{A} \times \mathcal{T}$ is the universe of events. A trace is a finite sequence of events $\sigma=\langle e_1, e_2, ..., e_n \rangle \in \mathcal{E}^*$ with size $n \in \mathbb{N}$ such that for each $1 \leq i < n$, $\pi_{\mathcal{C}}(e_i)=\pi_{\mathcal{C}}(e_{i+1}) \wedge \pi_{\mathcal{T}}(e_i) \leq \pi_{\mathcal{T}}(e_{i+1})$. An event log L is a set of traces such that each trace belongs to a different case. $\mathcal{L}$ is the universe of event logs. $|L|$ is the number of traces in event log L. In addition, we define $cf : \mathcal{L} \rightarrow \mathcal{B}(\mathcal{A}^*)$ as a function that extracts the control flow of L, i.e., the multiset of traces projected on activities such that $cf(L)=[\langle \pi_{\mathcal{A}}(e_1), ..., \pi_{\mathcal{A}}(e_n) \rangle | \langle e_1, e_2, ..., e_n \rangle \in L]$.*

In addition to the control flow and time dimension, other dimensions could be assigned to cases from other sources of information. We use $\kappa_L : L \rightarrow \mathbb{R}$ to show a case-level indicator. Case-level indicators assign a numerical value $\kappa_L(\sigma)$ to each trace $\sigma \in L \in \mathcal{L}$. If the context is clear, we drop L from the notation $\kappa_L(\sigma)$.

Definition 2. *(Ordering function) Let $L \in \mathcal{L}$ be an event log and κ be a case-level indicator that assigns a value to each case in L. We define $rank_{\kappa}(L) = \langle \sigma_1, \sigma_2, ..., \sigma_{|L|} \rangle$ such that $L=\{\sigma_1, \sigma_2, ..., \sigma_{|L|}\}$ and for $1 \leq i < j \leq |L| : \kappa(\sigma_i) \leq \kappa(\sigma_j)$.*

For example, $L = \{\langle (c_1, a, 13), (c_1, b, 23) \rangle, \langle (c_2, a, 14), (c_2, b, 16), (c_2, c, 20) \rangle, \langle (c_3, a, 17), (c_3, b, 20), (c_3, c, 35) \rangle\}$ is an event log with $cf(L)=[\langle a, b, c \rangle^2, \langle a, b \rangle^1]$. Consider κ as a function that calculates the duration of cases, therefore, $\kappa = \{(\sigma_{c_1}, 10), (\sigma_{c_2}, 6), (\sigma_{c_3}, 18)\}$ and $rank_{\kappa}(L) = \langle \sigma_{c_2}, \sigma_{c_1}, \sigma_{c_3} \rangle$.

Definition 3. *(Stochastic Language) Given the universe on activities $\mathcal{A}$, $f : \mathcal{A}^* \rightarrow [0,1]$ is a stochastic language iff $\sum_{s \in \mathcal{A}^*} f(s)=1$. $\mathcal{F}$ is the universe of stochastic languages.*

Definition 4. *(Stochastic Language of an Event Log) Let $L \in \mathcal{L}$ be an event log. $stoch : \mathcal{L} \rightarrow \mathcal{F}$ is a function that extracts the stochastic language of an event log such that $stoch(L) = \{(s,p)|s \in cf(L) \wedge p = \frac{cf(L)(s)}{|L|}\}$.*

The earth mover's distance calculates the distance between two stochastic languages. We use this distance to compare the control flow in two event logs.

Definition 5. *(The earth mover's distance) Let $L \in \mathcal{L}$, and $L' \in \mathcal{L}$ be two event logs, $\delta : \mathcal{A}^* \times \mathcal{A}^* \rightarrow [0,1]$ be a trace distance function, and $r : L \times L' \rightarrow [0,1]$ be a function that indicates the movement of frequency between two event logs. $\mathcal{R}$ is the universe of all reallocation functions. The earth mover's distance between L and L' is defined by $EMD(L,L') = \min_{r \in \mathcal{R}} \sum_{s \in cf(L)} \sum_{t \in cf(L')} r(s,t).\delta(s,t)$ such that $\forall s \in cf(L) : \sum_{t \in cf(L')} r(s,t) = stoch(L)(s)$, $\forall t \in cf(L') : \sum_{s \in cf(L)} r(s,t) = stoch(L')(t)$, and $\forall s \in cf(L), \forall t \in cf(L') : r(s,t) \geq 0$.*

We use the Levenshtein distance δ to calculate the distance between the traces. The calculation of the efficient reallocation amounts, i.e., r in Def. 5 is solved as a linear programming problem. For more details, we refer to [9].

4 Process Variant Identification Framework

As illustrated in Fig. 1b, the framework proposed in this paper consists of two main parts, a control flow change detection over the range of a case-level indicator and a segmentation and comparison framework. Unlike concept drift frameworks which focus on time dimension, we identify changes in control flow across continuous dimensions. Our framework does not generate a large feature set that might overlook certain types of changes. Instead, it leverages the earth mover's distance to effectively capture the variability in control flow.

4.1 Control Flow Change Detection

Our proposed method consists of a bucketing step and then moving a sliding window. In Fig. 2a, three different event logs, i.e., L_1, L_2, and L_3 are illustrated. Each of the event logs has 150 cases and the cases are ordered based on a case-level indicator. A bucketing strategy is used to divide the event logs into 15 buckets each containing 10 traces. The buckets are shown as patterned boxes such that different patterns show that the traces are different. In this example, two different window sizes $w_1{=}1$ and $w_2{=}3$ are used. Considering event log L_1 and window size w_1, we start from $i = 1$ and each time we move the central point one unit to the right up to $i = 14$ and compare w_1 bucket on the left-hand side of the central point to w_1 bucket on the right-hand side of it using the earth mover's distance. Comparative results for some combinations of the event logs and time windows are illustrated in Fig. 2b using a color range proportional to the difference level. Next, we formally define buckets and sliding windows.
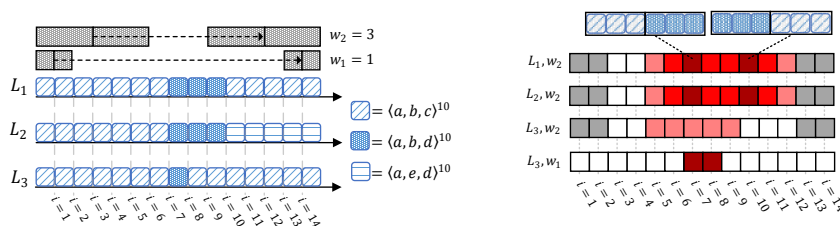
(a) Event logs L_1, L_2, and L_3 with different types of changes in the control flow.

(b) Visualised *ldist* value with moving the sliding window.

Fig. 2: Some examples illustrating how the framework works considering three different event logs, $b = 15$, and two time windows $w_1 = 1$ and $w_2 = 3$.

Definition 6. *(Buckets) Let $L \in \mathcal{L}$ be an event log, κ be a case-level indicator, $\sigma_i = rank_\kappa(L)(i)$, and $b \in \mathbb{N}^{[2,|L|]}$ be the number of buckets given by the user. For the sake of simplicity, we assume that $|L|$ is divisible by b. Then, $l = \frac{|L|}{b}$ is the number of cases in each bucket such that $B_i = \langle \sigma_{(i-1).l+1}, ..., \sigma_{i.l} \rangle$ for $1 \leq i \leq b$.*

Definition 7. *(Left and Right Windows) Let $L \in \mathcal{L}$ be an event log, κ be a case-level indicator, $\sigma_i = rank_\kappa(L)(i)$, $b \in \mathbb{N}^{[2,|L|]}$ be the number of buckets given by the user, and $w \in \mathbb{N}^{[1,\frac{b}{2}]}$ be a window size parameter. Considering a window size w and $i \in \{w, ..., b-w\}$, we can create a left window $W_{w,i}^l = \{\sigma \in B_j | i - w < j \leq i\}$ and a right window $W_{w,i}^r = \{\sigma \in B_j | i < j \leq i + w\}$.*
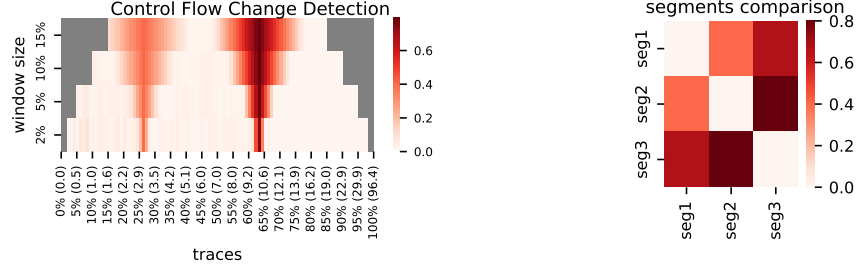
To perform a comparative analysis, we move the sliding window over the whole range of κ. We need at least w buckets on the left window. Therefore, we start from $i = w$ and each time move the sliding window for one bucket. We repetitively continue until we reach a point where the number of buckets on the right window is equal to the window size, i.e., $i = b - w$.

Definition 8. *(Local Distance Function) Let $L \in \mathcal{L}$ be an event log, κ be a case-level indicator, $b \in \mathbb{N}^{[2,|L|]}$ be the number of buckets given by the user, and $w \in \mathbb{N}^{[1,\frac{b}{2}]}$ be a window size parameter. We define the local distance function $ldist_{L,\kappa,w,b} : \{w, ..., b-w\} \to [0,1]$ such that $ldist_{L,\kappa,w,b}(i) = EMD(W_{w,i}^l, W_{w,i}^r)$. If the context is clear, we show $ldist_{L,\kappa,w,b}$ as $ldist$.*

In Fig. 2b, some analysis is performed based on different event logs and different window sizes. Using the event log L_1 and the window size w_2, from $i = 5$ the change in behavior is detectable by calculating the earth mover's distance between the left and right windows. The earth mover's distance value is visualized with colors, i.e., darker color shows a higher distance. After shifting the sliding window, the maximum difference between the left and right windows is observed at $i = 7$ and $i = 10$. A similar pattern in changing $ldist$ value is observed using L_2 and w_2. However, the introduced framework does not give us any clue whether the control flow behavior before $i = 7$ and after $i = 10$ are similar or different. Using L_3 and the smaller window w_1, it is observed that the change in behavior is only observed sharply at $i = 7$ and $i = 10$. Using the larger window w_2, the change in behavior started to affect $ldist$ from $i = 5$ but with shifting the sliding window, it cannot exactly identify the point in which the behavior is changed. Therefore, the larger window size is more robust against noise but may miss some important change points.

The running example introduced in Fig. 1a is simulated using the CPN tools and the generated event log is used to explain how the framework works. This event log consists of 10,000 cases and 31 trace variants[1]. The cases are ordered based on their risk score value. In Fig. 3a, the results using $b = 100$, i.e., 1 % of the cases in each bucket and different window sizes $w \in \{2, 5, 10, 15\}$ are shown. The numbers in the parenthesis show the raw risk score values. The experiment is

---

[1] `https://github.com/aliNorouzifar/process-variants-identification/blob/main/event%20logs/test.xes`

(a) Results of applying the introduced framework using the simulated event log with $b{=}100$ and $w \in \{2, 5, 10, 15\}$.

(b) Pairwise comparison between the segments.

Fig. 3: Process variant identification framework for the claim handling process.

repeated for different window sizes. The colors show how high the earth mover's distance is between the left and right windows.

4.2 Segmentation and Pairwise Comparison

Based on the visualizations shown in Fig. 2, the control flow behavior may change several times through the range of the indicator but with each change, we only know that w buckets on the left are different from the w buckets on the right. Therefore, we extend the proposed framework to make it more applicable to obtain global insights. The idea is to use the peaks in the *ldist* values based on specific b and w parameters to perform a segmentation. Each time we observe a peak in *ldist*, we generate a segment consisting of the cases from the previous change point to the current change point. Then, it is possible to compare non-adjacent segments with the earth mover's distance measure.

Definition 9. *(Change Point in Control Flow) Let $L \in \mathcal{L}$ be an event log, κ be a case-level indicator, $b \in \mathbb{N}^{[2,|L|]}$ be the number of buckets given by the user, and $w \in \mathbb{N}^{[1,\frac{b}{2}]}$ be a window size parameter. $\theta \in [0,1]$ is a user-defined threshold to check whether a high value of ldist is significant. $p \in [w, b-w]$ is a change point in the control flow behavior if $ldist(p) \geq \theta$, $ldist(p-1) \leq ldist(p)$ if $p \in (w, b-w]$, and $ldist(p+1) \leq ldist(p)$ if $p \in [w, b-w)$.*

Definition 10. *(Segments) Let $P = \langle p_1, ..., p_{|P|} \rangle$ be the ordered sequence of change points in ldist function such that $\forall i, j \in \{1, ..., |P|\} : p_i \leq p_j$ iff $i < j$. Considering $|P|$ as the number of peaks, we can generate $|P|+1$ segments which we refer to as seg_1 to $seg_{|P|+1}$ such that $seg_1 = \{\sigma \in B_x | x \leq p_1\}$, $seg_i = \{\sigma \in B_x | p_{i-1} < x \leq p_i\}$ for $i \in [2, |P|]$, and $seg_{|P|+1} = \{\sigma \in B_x | p_{|P|} < x\}$.*

Considering $w = 10$ in Fig. 3a and $\theta = 0.1$, *ldist* has two peaks in $i = 26$ (*risk score*=3) and $i = 63$ (*risk score* = 10). We can use heatmaps to compare the resulting segments pairwise to check if non-adjacent segments have similar

control flows. Using the identified peaks, the whole event log is divided into three segments, and the segments are compared in Fig. 3b.

Next, we can hierarchically merge similar segments in case the control flow is not significantly different with regard to the user-defined difference threshold θ. We recursively merge two segments with the minimum *ldist* if this minimum distance is lower than θ, i.e., a user-defined threshold to check whether the distance is significant.

5   Case Study

The introduced framework is implemented and is publicly available[2]. To assess the effectiveness of our framework, we conducted a comprehensive case study in collaboration with UWV, the employee insurance agency of the Netherlands responsible for executing social security in case of unemployment and disability. UWV provided real-life event data for this case study. We worked closely with experts from the agency, who provided invaluable guidance and insights. Their expertise enabled us to align our research with practical, real-world scenarios, ensuring that our algorithms are usable in real business contexts.

5.1   UWV event log

UWV has over 18000 employees and several branches all over the Netherlands. One of the main processes of this governmental sector is investigated in this paper. The duration of the cases in this process has been of particular interest. UWV wants to know how the process is changed concerning the duration of the cases and whether we can find the change points in the duration. The duration of cases varies in the range of 1 day to 575 days. The event log has 144,096 cases, 1,026 variants, 29 activities, and 1,316,128 events. Among all the cases, 5,449 cases (3.8% of the cases) are rejected and 138,647 cases (96.2% of the cases) are accepted.

The normative BPMN model shown in Fig. 4 represents a claim-handling process at UWV. First, a claim from a customer is received. Then, either the claim is accepted or blocked. A blocked claim indicates either some information needs to be checked or corrected after which the claim is accepted, or the claim is blocked and then immediately rejected. After a rejected claim, an objection can be received by customers if they do not agree with the decision. The handling of this objection is out of the scope of this process model. After an accepted claim, which has received one or at most three payments, an objection can be received. This is due to customers who in hindsight find that they do not need the payments. A claim withdrawal process is started that results in repayment of the total sum of received benefits. In case the customer is still entitled to one or more payments, the *Block Claim* activity is executed. This prevents any new payments from being automatically made. This normative model has an alignment fitness of 99.3% with respect to the event log [3].
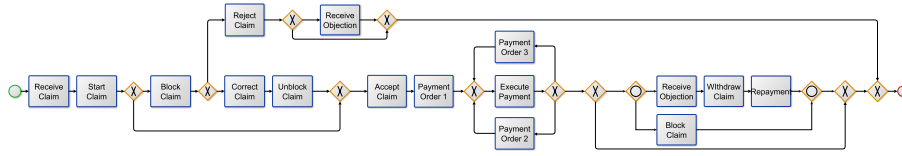
---
[2] https://github.com/aliNorouzifar/process-variants-identification

Fig. 4: The normative BPMN model representing the investigated UWV claim handling process.

5.2 Process variant analysis using UWV event log

The first experiment is performed using the complete event log and the results are shown in Fig. 5. In this experiment, $b=100$ and the window sizes 2, 5, 10, and 15 are considered. The comparative results are illustrated in Fig. 5a which shows that the cases with a very short duration or very long duration are different from other cases. Considering $\theta=0.1$, $w=5$, and i referring to the central point of the sliding window, two peaks are observed at $i=3$ (17 days) and $i=95$ (78 days) which can be used to generate three segments, i.e., [0,17] days, [18,77] days and [78,575] days. In Fig. 5b, the three generated segments are compared to each other using the earth mover's distance.

The number of cases in each segment and their alignment fitness with respect to the normative model is reported in Table. 1. All the segments have a high fitness value implying that the observed behavior in the segments fits the normative model. According to the replay results, Fig. 6 shows the normative BPMN model where the parts of the model that cover each of the segments are highlighted. The colors are proportional to the frequencies and the transition labels are colored as gray if the frequency is lower than 5%. For segment 1 in Fig. 6a, the activities *Receive Claim*, *Start Claim*, *Block Claim*, *Reject Claim* and *Receive Objection* are highlighted. *Segment 1* represents the claims that are rejected and optionally UWV receives an objection to the decision. *Segment 2* shows claims that are accepted, i.e., the customer receives three payments and then the process finishes. Finally, *segment 3* represents again accepted cases. However, after the payments an objection is received, and optionally a block
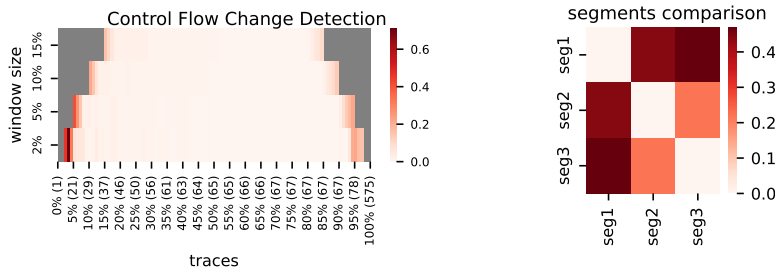

(a) Control flow change detection with $b=100$ and $w \in \{2, 5, 10, 15\}$.

(b) Segmentation and pairwise comparison.

Fig. 5: Applying the introduced framework to the complete UWV event log.

Table 1: Using alignment fitness as a conformance checking metric to check whether the event logs fit the process model or not.

| Event log		Number of traces		Alignment fitness	
UWV complete log	segment 1: [0,17] days		4323		99.6%
	segment 2: [18,77] days	144096	132572	99.3%	99.7%
	segment 3: [78,575] days		7201		93.4%
UWV rejected claims	segment 1: [0,15] days		3888		99.7%
	segment 2: [16,83] days	5449	1350	98.4%	97.3%
	segment 3: [84,550] days		211		81.7%

is executed. For these claims, the customer withdraws the claim. The customer then pays the received sum back to UWV. Blocking is done to prevent new payments from being made while the claim withdrawal process has not finished yet.
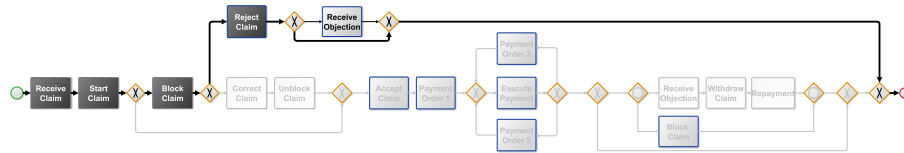
(a) Segment 1 in the complete event log with duration in the range [0,17] days.
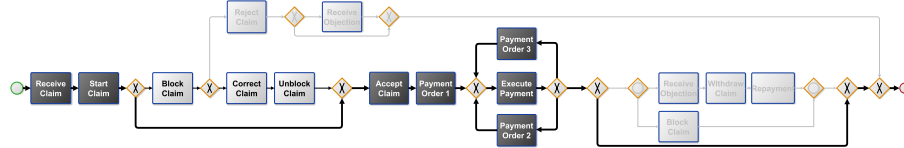
(b) Segment 2 in the complete event log with duration in the range [18,77] days.
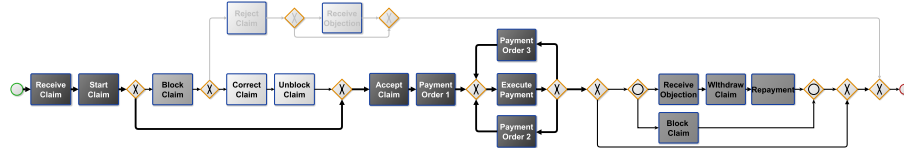
(c) Segment 3 in the complete event log with duration in the range [78,575] days.

Fig. 6: The normative BPMN model is highlighted based on the frequency of the transitions in replaying the segments from the complete event log experiment.

5.3   A deeper analysis considering rejected cases

The extracted segments are highly correlated with the process outcome, i.e., the rejection or acceptance of a claim. However, the correlation between the extracted segments with specific duration is less clear. Another experiment using the 5,449 rejected cases is performed to get more understanding of the relation

between duration ranges and specific process variants[3]. The alignment fitness of the rejected cases is 98.04% with respect to the normative model.

An overview of the results for the rejected cases is shown in Fig. 7. Based on Fig. 7a considering $w = 2$ and $\theta=0.1$, four segments are found, with duration periods of [0,13] days, [14,15] days, [16,83] days, and [84,550] days respectively. Segments 1 and 2 are merged into one segment, with a duration period of [0,15] days, since the distance between them is lower than $\theta=0.1$. This can be observed in the heatmap in Fig. 7b. Segments 1 and 2 have very similar colors when compared to each other in comparison to the other segments.
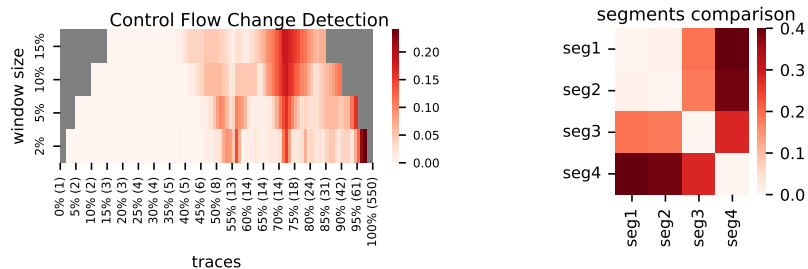
(a) Control flow change detection with $b=100$ and $w\in\{2,5,10,15\}$.

(b) Segmentation and pairwise comparison.

Fig. 7: Applying our framework to the rejected cases in the UWV event log.

The number of cases in each segment and alignment fitness values of the identified segments are shown in Table 1. These results show that segment 3 does not fit the normative model well with an alignment fitness value of 81.7%. Fig. 8 contains the highlighted normative model for each of the three segments. Fig. 8a shows the first segment with a duration period of [0,15] days and contains cases that are rejected and no objection is received. This segment represents customers who file a claim even though they know that they most likely will not be entitled. In the second segment, with a duration period [16,83] days, in Fig. 8b cases are described that are rejected and some also have an objection. Finally, the third segment, with a duration period of [84,550] days, in Fig. 8c consists of claims that are first accepted and end as rejection with a full repayment, or claims that are first rejected and are accepted after an objection is received and granted.

6   Conclusion

In this study, we delved into the concept of segmenting a continuous dimension concerning changes in control flow. While our framework primarily focuses on identifying segments with differing control flows, it may not directly imply the

---

[3] The analysis of accepted cases is explained in the supplementary material https://github.com/aliNorouzifar/process-variants-identification/blob/main/supplementary%20material/supplementary%20material.pdf.
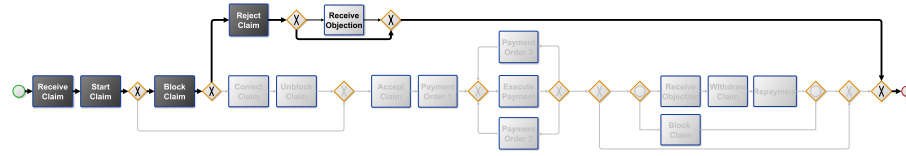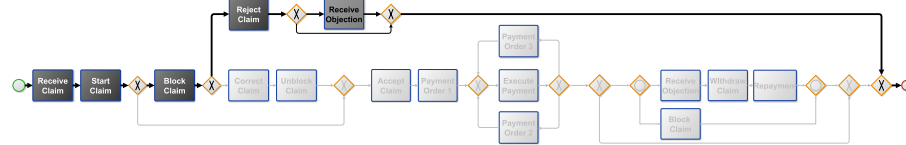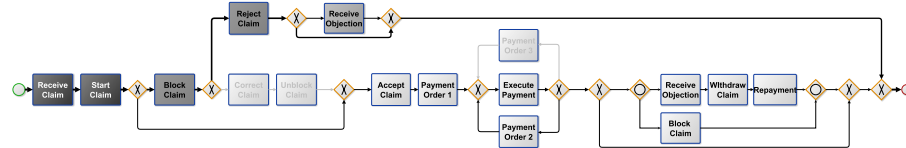
(a) Segment 1 in the rejected cases event log with duration in the range [0,15] days.

(b) Segment 2 in the rejected cases event log with duration in the range [16,83] days.

(c) Segment 3 in the rejected cases event log with duration in the range [84,550] days.

Fig. 8: The normative BPMN model highlighted based on the frequency of the transitions in replaying the segments from the rejected event log experiment.

desirability or undesirability of individual cases. For instance, with a dimension like duration, cases within segments featuring either very short or very long duration could be associated with undesirability or efficiency. This approach holds promise for extension, particularly in identifying and labeling cases as undesirable, taking the broader process context into account. The interconnections between different dimensions may also play a role in the changes in behavior which is highly relevant for future investigations. Considering them may lead to some interesting analyses. For instance, the duration of cases could correlate with the workload of the process, potentially generating delayed yet normal cases during high workload periods. Considering only one dimension may not show whether cases are normal or problematic. The output of our framework can be used as input for various process mining tasks like process comparison, outcome prediction, and process discovery using desirable and undesirable cases.

References

1. Back, C.O., Simonsen, J.G.: Comparing trace similarity metrics across logs and evaluation measures. In: Advanced Information Systems Engineering, CAiSE 2023, Proceedings. Lecture Notes in Computer Science, vol. 13901, pp. 226–242. Springer (2023)
2. Brockhoff, T., Uysal, M.S., van der Aalst, W.M.P.: Time-aware concept drift detection using the earth mover's distance. In: 2020 2nd International Conference on Process Mining (ICPM). pp. 33–40. IEEE (2020)

3. Carmona, J., van Dongen, B.F., Solti, A., Weidlich, M.: Conformance Checking - Relating Processes and Models. Springer (2018)
4. Chapela-Campa, D., Mucientes, M., Lama, M.: Understanding complex process models by abstracting infrequent behavior. Future Gener. Comput. Syst. **113**, 428–440 (2020)
5. Chesani, F., Francescomarino, C.D., Ghidini, C., Grundler, G., Loreti, D., Maggi, F.M., Mello, P., Montali, M., Tessaris, S.: Shape your process: Discovering declarative business processes from positive and negative traces taking into account user preferences. In: Enterprise Design, Operations, and Computing, EDOC 2022, Proceedings. Lecture Notes in Computer Science, vol. 13585, pp. 217–234. Springer (2022)
6. De Leoni, M., van der Aalst, W.M.P., Dees, M.: A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs. Information Systems **56**, 235–257 (2016)
7. Hompes, B., Buijs, J., van der Aalst, W.M.P., Dixit, P., Buurman, J.: Discovering deviating cases and process variants using trace clustering. In: Proceedings of the 27th Benelux Conference on Artificial Intelligence (BNAIC). pp. 5–6 (2015)
8. Leemans, S.J., Shabaninejad, S., Goel, K., Khosravi, H., Sadiq, S., Wynn, M.T.: Identifying cohorts: Recommending drill-downs based on differences in behaviour for process mining. In: Conceptual Modeling: 39th International Conference, ER 2020, Proceedings 39. pp. 92–102. Springer (2020)
9. Leemans, S.J., Syring, A.F., van der Aalst, W.M.P.: Earth movers' stochastic conformance checking. In: Business Process Management Forum: BPM Forum 2019, Proceedings 17. pp. 127–143. Springer (2019)
10. Maaradji, A., Dumas, M., Rosa, M.L., Ostovar, A.: Detecting sudden and gradual drifts in business processes from execution traces. IEEE Trans. Knowl. Data Eng. **29**(10), 2140–2154 (2017)
11. Norouzifar, A., van der Aalst, W.M.P.: Discovering process models that support desired behavior and avoid undesired behavior. In: SAC '23: The 38th ACM/SIGAPP Symposium on Applied Computing (2023)
12. Sato, D.M.V., Freitas, S.C.D., Barddal, J.P., Scalabrin, E.E.: A survey on concept drift in process mining. ACM Comput. Surv. **54**(9), 189:1–189:38 (2022)
13. Taymouri, F., La Rosa, M., Dumas, M., Maggi, F.M.: Business process variant analysis: Survey and classification. Knowledge-Based Systems **211**, 106557 (2021)
14. Teinemaa, I., Dumas, M., Rosa, M.L., Maggi, F.M.: Outcome-oriented predictive process monitoring: Review and benchmark. ACM Trans. Knowl. Discov. Data **13**(2), 17:1–17:57 (2019)
15. Weerdt, J.D., vanden Broucke, S.K.L.M., Vanthienen, J., Baesens, B.: Active trace clustering for improved process discovery. IEEE Trans. Knowl. Data Eng. **25**(12), 2708–2720 (2013)
16. Yeshchenko, A., Ciccio, C.D., Mendling, J., Polyvyanyy, A.: Visual drift detection for event sequence data of business processes. IEEE Trans. Vis. Comput. Graph. **28**(8), 3050–3068 (2022)
17. van Zelst, S.J., Cao, Y.: A generic framework for attribute-driven hierarchical trace clustering. In: Business Process Management Workshops - BPM 2020 International Workshops, Revised Selected Papers. Lecture Notes in Business Information Processing, vol. 397, pp. 308–320. Springer (2020)