Toward a New Generation of Log Pre-processing Methods for Process Mining

Paolo Ceravolo[1], Ernesto Damiani[2], Mohammadsadegh Torabi[1], and Sylvio Barbon Junior[3]

[1] Università degli Studi di Milano
via Bramante 65, Crema, Italy
paolo.ceravolo@unimi.it | mohammadsadegh.torabi@studenti.unimi.it
[2] Khalifa University
PO Box 127788, Abu Dhabi, UAE
ernesto.damiani@kustar.ac.ae
[3] Londrina State University
Rod. Celso Garcia Cid, 445 - Londrina, Brazil
barbon@uel.br

Abstract. Real-life processes are typically less structured and more complex than expected by stakeholders. For this reason, process discovery techniques often deliver models less understandable and useful than expected. In order to address this issue, we propose a method based on statistical inference for pre-processing event logs. We measure the distance between different segments of the event log, computing the probability distribution of observing activities in specific positions. Because segments are generated based on time-domain, business rules or business management system properties, we get a characterisation of these segments in terms of both business and process aspects. We demonstrate the applicability of this approach by developing a case study with real-life event logs and showing that our method is offering interesting properties in term of computational complexity.

Keywords: Process Mining, Event-log Clustering, Pre-processing, Lightweight Trace Profiling

1 Introduction

The well-known idiomatic expression "garbage in, garbage out" applies well to Process Mining (PM), because significant results can be achieved only if the *event logs* fed into PM algorithms are good examples of execution for all the relevant variants in a business process [**?**].

This problem is already recognised by the literature and many contributions underline that before running process discovery it is required to pre-process event logs [**?**,**?**]. Clustering is considered one of the most relevant pre-processing tasks as grouping similar event logs can radically reduce the complexity of the

discovered models [?,?,?,?,?][4]. Despite this attention, the methods proposed in the literature are only partially tailored to the specific needs of Business Process Management (BPM) [?], where business goals and rules [?] are tailored on each specific business process then monitoring or discovery should be tailored accordingly [?].

Few existing process mining techniques are equipped with means for uncovering differences among event logs. Moreover, with the notable exception of [?], little attention has been devoted to the development of a comprehensive method for coping with the entire work-flow that guides pre-processing tasks. This work-flow includes at least the following steps: (i) *characterisation of events logs*, (ii) *computation of a similarity measure* and finally (iii) *evaluation on the business relevance* of the divergences or convergences of the characteristics considered. These tasks cannot be considered in isolation and multiple iterations over them may be required to get significant results.

Differently from the currently adopted log pre-processing practises, the approach we propose in this paper introduces the notion of segment that is a sub set of the event log that conforms to some specific business goals or business rules. Statistical inference-based analysis allows to characterise the distribution of activities in segments, providing an explanation of their similarities or dissimilarities. More specifically the paper is organized as follows: we start introducing the related works in Sect. 2. We then present an overview on our method in Sect. 3[5]. In particular, in Sect. 3.1 we provide preliminary definitions and explain how event logs can be segmented; in Sect. 3.2 we introduce a new trace profiling method that can be exploited in comparing and clustering traces; in Sect. 3.3 we illustrate distance metrics based on inferential statistics; in Sect. 3.4 we discuss how to use our results to characterise segments and evaluate if they are suitable to input process discovery. In Sect. 4 we demonstrate the applicability of our method using a case study with real-life event log from an Italian manufacturing company[6]. In Sect. 5 we compare our method to the state of the art via a time complexity analysis and finally in Sect. 6 we draw our conclusions.

2 Related Work

Just like Data Transformation [?] and Data Cleansing [?], Trace Clustering [?] is a crucial step in pre-processing event logs, as it can radically reduce the inherent complexity of discovered models. Song et al., in [?], present an introduction to

[4] Some works, such as for instance [?], define as "Clustering" the identification of similar activities, this is also a pre-processing task relevant to our discussion, however in this paper we are using "Clustering" for referring uniquely to the process of segmenting event logs.

[5] The Python implementation of the algorithms adopted to implement and test our method is available at `http://www.uel.br/grupo-pesquisa/remid/wp-content/uploads/LightPMClustering.rar`

[6] The event log is available at `http://www.uel.br/grupo-pesquisa/remid/wp-content/uploads/EventLogDatasetAnon.csv`

trace clustering algorithms with trace profiling. A profile is a set of related items that describe a trace from a specific perspective. These perspectives usually rely on derived information, such as the number of events in a trace or the resources consumed during execution. A profile with n items is a function, which assigns to a trace a vector with n elements. Encoding a trace into a vector space model makes possible to compute distance metrics and perform cluster analysis.

In [**?**], authors use a trace clustering approach based on edit distance[7], where profiles are obtained by listing the activities into a trace (bag-of-activities). This is a straightforward approach that offers linear computational complexity when computing a distance measure between traces, but loses all information on the trace structure.

To incorporate information about trace structure, it is possible to adopt contextual approaches. These approaches generate vectors using k-grams [**?**], i.e. representing each activity in the trace as a sub-sequence of length k. Even though it has been shown that techniques that take into account context perform better than those that do not, the high complexity of k-gram, $\mathcal{O}(n^k)$, is an obstacle respecting most of the state of art methods with linear complexity.

Recent research focuses on extracting multiple trace profiles to exploit multi-criteria clustering techniques. In [**?**] the authors proposed a framework to deal with the more general correlation problem by a tool that merges previous approaches in the literature. Appice and Malerba, in [**?**], proposed co-training clustering as a pre-processing step. The output is a trace clustering pattern, obtained by clustering the traces across multiple profiles inputted. The co-training idea is based on iterative modification of a similarity matrix extracted from the trace profile. The time complexity of such an algorithm depends on the cost of computing the similarity and clustering matrices which are respectively $\mathcal{O}(n^2M)$ and $\mathcal{O}(d)$, where n is the number of traces, M the average number of features per trace profile and d the cost of the distance-based algorithm used.

When computing the similarity between two activities most methods do not deal with semantics, e.g. cannot capture tasks that are expressed using different abstraction levels but refer to the same business activity. Chen et al. proposed in [**?**] a method which can address semantic aspects as well as structural features of the event log. Their pre-processing method is based on k-means clustering, whose cost is $\mathcal{O}(n^{dk+1})$ over vectors that encode both structure and semantic features, where n is the number of traces, d the cost of distance function and k the number of compared traces. Structure information is derived from control-flow representations such as loops, branches and sequences. It is however important to note that the proposed solution can extract control-flow and semantics features only if a deep pre-processing analysis is performed, thus the challenges we outlined in the introduction are moved rather than solved.

[7] The edit distance between two strings is the minimum number of operations required to transform one string into the other.

3 Overview on the Proposed Method

An high-level overview of the method proposed in this paper is shown in Fig. 1. Our starting point is an event log that collects a set of cases, i.e. instances of business process execution. To select sets of cases that can meaningfully be fed to process discovery we propose a method organised in four steps. A criteria for *segmenting the event log* is first identified; segments are then *represented based on a trace profile*, where a trace is a unique sequence of events generated in executing a business process, which is adopted to *compute a similarity measure*; finally the adopted segmentation is *assessed and characterised*. In the following we describe each step providing a formalisation of the operations implemented and related examples to clarify the details.

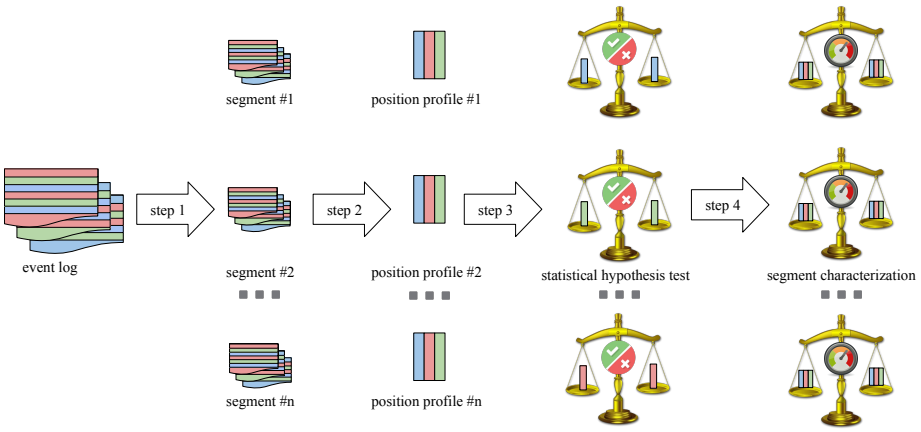

Fig. 1. Overview of method proposed in this research

3.1 Step 1: Segmenting the Event Log

The first step is splitting the event log into group of cases called segments. We rely on the event log description standard proposed by the IEEE Task Force on Process Mining. The eXtensible Event Stream (XES) [**?**] defines a grammar for a language capturing information systems' behaviors. In this framework, an event stream describes a set of events that can be ordered in a sequence using their execution timestamps. More specifically an event can be defined according to Definition 1.

Definition 1. *Event. An event is a quadruple* $e = (c, a, r, t) \in \mathcal{E}$*, denoting the occurrence of an activity* a *in a case* c*, using the resource* r *at time* t*. The event universe can be indicated as the Cartesian product:* $\mathcal{E} = \mathcal{C} \times \mathcal{A} \times \mathcal{R} \times \mathcal{T}$*.*

As stated in Definition 2, each event reports on the execution of an activity within a specific instance of the business process, usually called *case*.

Definition 2. *Case. Let $\mathcal{E}$ be a finite set of events. A case $\sigma \in \mathcal{E}*$ is a finite sequence of events belonging to $\mathcal{E}$ and related to a same process execution.*

All cases characterised by the same sequence of events are represented by the same trace, as stated in Definition 3.

Definition 3. *Trace. Let $\mathcal{A}$ be a finite set of activities. A trace $\theta \in \mathcal{A}*$ is a finite sequence of activities belonging to $\mathcal{A}$.*

Process Mining algorithms interpret an event log as a multi-set of traces and infer models comparing these sequences of events. We argue that this notion is not necessarily capturing the business goals of the organisation in addressing a case. For this reason, our pre-processing analysis starts by segmenting the event log base on business goals. This way, segments can be compared to relevant business requirements and further steps of refinement can be oriented based on a specific maximisation criteria. Given our definition of case, we can now formalise the definition of segment, according to Definition 4.

Definition 4. *Segment. Given n cases, a segment s is a union of cases: $s = (c_1 \cup c_2 \cup, \ldots, c_n)$ where s is a subset of an Event Log: $s \subseteq \mathrm{L}$.*

A variety of criteria can be used to segment event logs [**?**], including temporal constraints, case type, business rule compliance, performance result, resources involved in the execution and others. In the rest of this paper we will generically refer to these criteria as business rules. From an operational point of view, a segment can be identified by a query over a set of predicates that can be joined with one of the elements composing an event, as proposed in [**?**]. Table 1 shows an excerpt of a real-life event log. Segmenting by the values in the field `Customer` we get three segments: $s_1 = \{Case1, Case5\}$, $s_2 = \{Case2, Case4, Case6\}$, and $s_3 = \{Case3\}$.

3.2 Step 2: Trace Profiling

Following [**?**], profiling a log can be described as the aggregation in a vector of a set of measures on the events composing a trace. These vectors can be used to calculate the distance between any two traces, using a suitable distance metric. In this work, we are proposing a new method for profiling traces that can be extended to segments and that offers a good trade-off between computational complexity and context aware encoding, as discussed in Sect. 5.

The basic idea is that the structure of the event log is reduced to a list of activities and each activity has a vector of positions. This vector is defined as a list of ordinal positions with corresponding frequency. With this definition, the set of elements representing an event is extended from a binary relation {case $\times$ activity} to a ternary relation {case $\times$ activity $\times$ position}, as stated in Definition 5. Nevertheless, the representation format is kept bi-dimensional by creating an element in the vector for each couple { activity $\times$ position}.

Table 1. An example of real-life event log.

Case ID	Activity	Customer	Case ID	Activity	Customer
1	process creation	Gng inc.	4	process creation	MAS spa.
1	configuration manager	Gng inc.	4	configuration manager	MAS spa.
1	weight	Gng inc.	4	me fabrication checker	MAS spa.
1	m_p	Gng inc.	4	weight	MAS spa.
1	stress	Gng inc.	4	stress	MAS spa.
1	me assembly checker	Gng inc.	4	m_p	MAS spa.
1	me fabrication checker	Gng inc.	4	me assembly checker	MAS spa.
1	design checker	Gng inc.	4	design checker	MAS spa.
1	design leader	Gng inc.	4	design leader	MAS spa.
2	process creation	MAS spa.	5	process creation	Gng inc.
2	configuration manager	MAS spa.	5	configuration manager	Gng inc.
2	me fabrication checker	MAS spa.	5	weight	Gng inc.
2	weight	MAS spa.	5	m_p	Gng inc.
2	stress	MAS spa.	5	me assembly checker	Gng inc.
2	m_p	MAS spa.	5	stress	Gng inc.
2	me assembly checker	MAS spa.	5	me fabrication checker	Gng inc.
2	design checker	MAS spa.	5	design checker	Gng inc.
2	design leader	MAS spa.	5	design leader	Gng inc.
3	process creation	Herw inc.	6	process creation	MAS spa.
3	configuration manager	Herw inc.	6	configuration manager	MAS spa.
3	weight	Herw inc.	6	me fabrication checker	MAS spa.
3	m_p	Herw inc.	6	weight	MAS spa.
3	me assembly checker	Herw inc.	6	stress	MAS spa.
3	stress	Herw inc.	6	m_p	MAS spa.
3	me fabrication checker	Herw inc.	6	me assembly checker	MAS spa.
3	design checker	Herw inc.	6	design checker	MAS spa.
3	design leader	Herw inc.	6	design leader	MAS spa.

Table 2. Position profile of event log in Table 1 using a simplified view where letters a-i in the trace represent activities in the following order: ['Process Creation', 'Configuration Manager', 'Weight', 'M_P', 'Stress', 'ME Assembly Checker', 'ME Fabrication Checker', 'Design Checker', 'Design Leader']

activity \ position	**p(1)**	**p(2)**	**p(3)**	**p(4)**	**p(5)**	**p(6)**	**p(7)**	**p(8)**	**p(9)**
a	6	0	0	0	0	0	0	0	0
b	0	6	0	0	0	0	0	0	0
c	0	0	3	3	0	0	0	0	0
d	0	0	0	3	0	3	0	0	0
e	0	0	0	0	4	2	0	0	0
f	0	0	0	0	2	1	3	0	0
g	0	0	3	0	0	0	3	0	0
h	0	0	0	0	0	0	0	6	0
i	0	0	0	0	0	0	0	0	6

Definition 5. *Position profile. A position profile is a triple* $apf = (a, p, f) \in \mathcal{E}$*, denoting the occurrence of an activity* a *at the position* p *with the frequency* f*. The event universe can be indicated as the Cartesian product:* $\mathcal{E} = \mathcal{A} \times \mathcal{P} \times \mathbb{N}$*.*

As an example, we convert Table 1 adopting the definition above and obtaining the representation shown in Table 2. Activities a,b,h and i are always at a fixed position, namely $1st$, $2nd$, $8th$, and $9th$. On the other hand, activity c is 3 times $3rd$ position and 3 times in $4th$ position. We call this table a *position profile*. More formally, a position profile can be encoded as an integer matrix via a two-dimensional function $f(x, y)$, where y is the temporal occurrence order of an activity x. The amplitude f of any pair (x, y) represents the number of occurrences of activity x at position y. Acyclic processes are represented by square binary matrices. In the case of processes containing cycles an activity can occurs in multiple positions[8].

Our encoding of the event log as an integer matrix allows us to perform several types of distance analysis, from simple matrix distance to neighbourhood evaluation. In other words, our matrix offers a novel computation-friendly representation for business processes event logs.

3.3 Step 3: Compute a Similarity Measure

Based on the matrix introduced above we are able to compute a degree of similarity between two different matrices. This similarity can distinguish different traces, segments or event logs, leading to results that naturally encode the control flow of a trace. Thus, given two matrices A and B, the similarity function

[8] Clearly, by generating segments the information on the control-flow encoded in matrices is aggregated using a compensative approach that can bias the comparisons. We plan to address this problem in future studies by using intra- and inter-segment similarity metrics.

can be defined as a generic norm $n(A, B)$. For example, one could simply subtract the number of occurrences reported in matrix A from the one in B, or compute edit distance [**?**] or cosine distance [**?**], as discussed in Sect. 2. However, in this work we want to propose an original approach for comparing trace profiles. The motivating idea is to identify a method that is not biased by a specific probability distribution. As discussed in [**?**], cosine similarity and other common similarity metrics are designed to work with normal distribution only, while this assumption is not made explicit in most of the approaches that adopt them.

Definition 6. *Hypothesis Testing. Let H_0 and H_{alt} denote the null and the alternative hypothesis respectively. Given two segments s_i and s_j, $\{s_i, s_j\} \subset s, \forall a \in \mathcal{A}$, a statistical test $ST(s_i, s_j, a)$ confirms H_0 when $\forall p$ from a it holds $(p_i, f_i) = (p_j, f_j)$; otherwise H_{alt} is confirmed.*

In Definition 6, ST is a statistical hypothesis test from parametric or nonparametric methods, a is an activity and p is its position, in accordance to Definition 5. In this research, after some trial-and-error on various tests, we focused on Jensen-Shannon divergence test and on a group of non-parametric statistical two-sample hypothesis tests based on correlation, namely the Spearman's and Kendall's rank correlation coefficient. The results returned by correlation test are expressed in term of a *p-value*, or calculated probability, that is the probability of finding the observed, or more extreme, values when H_0 is true. To make a decision of either accept or reject null hypothesis we should define a preset value called significance level or α for estimating the p-value. If p-value $< \alpha$ then we have sufficient evidence to reject H_0 and H_{alt} may be accepted. Otherwise, if p-value $> \alpha$, there is not sufficient evidence to conclude that the H_{alt} may be correct [9].

The Jensen-Shannon divergence is closely related to the KullbackLeibler distance (KL) which in turn can be approximated by the classic Chi-Square test. Given two vectors $\mathbf{A}$ and $\mathbf{B}$, $KL(\mathbf{A}, \mathbf{B})$ is calculated as $\sum a_i \ln \frac{a_i}{b_i}$. The Jensen-Shannon divergence compares two vectors $\mathbf{V}$ and $\mathbf{U}$ by averaging their probability distributions in a new vector $\mathbf{M} = \frac{1}{2}(\mathbf{V} + \mathbf{U})$. For each vector, it is computed a pair of values describing this divergence through the pair $KL(\mathbf{V}, \mathbf{M})$ and $KL(\mathbf{U}, \mathbf{M})$. To obtain a final distance metric, it is required to average the resulted divergence values and re-size the final result computing the square root. We can formalise this as the formula in Equation 1.

$$Jensen\text{-}Shannon\ Distance = \left(\frac{KL(\mathbf{V}, \mathbf{M}) + KL(\mathbf{U}, \mathbf{M})}{2} \right)^{\frac{1}{2}} \tag{1}$$

In Table 3 we compare two position profiles showing the results returned by the different metrics we considered. Note that for the Jensen-Shannon divergence the metric reports on the distance between two vectors, while when we use the

[9] Note that, when we do not reject H_0, it does not mean that H_0 is true. It means that the sample data have failed to provide sufficient evidence to cast serious doubt about the truthfulness of H_0.

Table 3. Table of position profiles of two segments and Jensen-Shannon distance (JS), Spearman's rank test (SR) and Kendall's rank test (KR)

activity \position	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	JS	SR (p-value)	KR (p-value)
act a in s4	15914	377	0	0	0	0	0	0	0	0	0	0.0899	0.0061	0.00105
act a in s8	5418	0	0	0	0	0	0	0	0	0	0			
act b in s4	1	4959	0	18	2357	8	306	1	0	0	0	0.3752	0.0986	0.04297
act b in s8	0	4022	0	0	0	0	0	0	0	0	0			
act c in s4	0	0	1060	756	516	1002	690	308	187	121	0	0.2330	0.0000	0.00009
act c in s8	0	0	833	558	359	239	159	95	2	0	0			
act d in s4	0	130	671	830	712	870	833	509	245	164	0	0.2277	0.0008	0.00185
act d in s8	0	0	433	576	459	319	241	217	0	0	0			
act e in s4	0	36	314	491	653	892	761	998	414	385	2	0.2498	0.0000	0.00018
act e in s8	0	0	174	228	330	404	329	772	8	0	0			
act f in s4	0	29	323	382	526	959	1209	525	400	315	1	0.2472	0.0000	0.00010
act f in s8	0	0	207	299	408	481	528	320	2	0	0			
act g in s4	0	26	162	329	351	448	683	748	344	363	0	0.2817	0.0014	0.00137
act g in s8	0	0	129	264	325	464	569	491	3	0	0			
act h in s4	0	1	725	543	602	468	447	274	0	0	0	0.0860	0.0001	0.00052
act h in s5	0	0	469	320	364	337	419	336	0	0	0			
act i in s4	0	0	50	0	11	307	18	1272	1739	230	1332	0.5396	0.0165	0.00555
act i in s5	0	0	0	0	0	1	0	14	2198	0	0			
compare(s4, s8)												0.2592	0.88	0.88

p-values the metric reports about the probability that the two vectors were generated by distributions sharing same characteristics. We do not provide here a full explanation on how the Spearman's and Kendall's rank correlation coefficients were calculated. The interested reader can refer to [**?**] for details. To return an overall value about the comparison of the two segments we adopt different approaches in case the ST is returning a p-value or not. When ST returns a distance measure we simply average the results obtained for each activity (third to last column in Table 3). While dealing with p-values, we compute an index stating how many times the calculated probability is less than the significance level α (second to last and last columns in Table 3, taking $\alpha = 0.01$).

3.4 Step 4: Characterise Segments

We have now a measure of the dissimilarity level of two probability distributions of the activities' position in segments. Using this dissimilarity metrics we can cluster segments as illustrated in Fig. 4. An *assessment criterion* for validating a segment is obtained by imposing a minimum dissimilarity value in comparison to the others segments in the event log. Clearly, this procedure can be applied with different metrics looking for at least one metric where the criterion is met. Any segment that is not compliant with assessment criteria need to be re-sized. Moreover, our method provides us with a measure of the specific contribution that each activity has provided in characterising a segment. For example, using the Jensen-Shannon divergence test we can identify activity b, g, and i as those that are introducing most divergence.

If our interest is not particularly related to measuring a distance we can exploit the non-parametric tests that provide us with a measure of the correla-

tion of two distributions. In modern use, "correlation" refers to a measure of a linear relationship between variables, while "measure of association" is usually referred to a measure of a monotone relationship between them. Two well-know examples that measure the latter type of relation are Kendall's tau and the Spearman rho metrics. Differently to the Jensen-Shannon distance, these metrics tell us if two distributions have a similar trend, without measuring a precise distance on frequencies. We can, in other words, detect traces or segments that are similar because the shape of their probability distribution even in presence of different absolute values. Fig. 2 shows the difference between monotonic and non-monotonic relation.

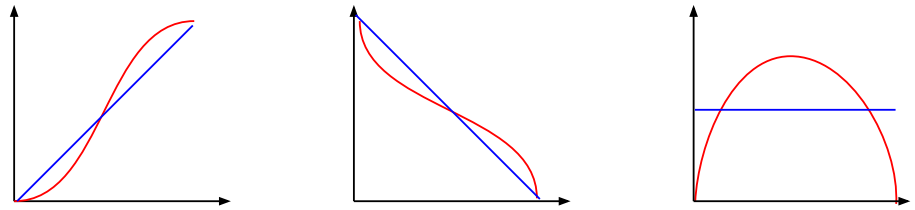

Fig. 2. A comparison between a monotonic and non-monotonic relationship (red color lines). From left to right: increasing monotonic, decreasing monotonic and non-monotonic relationship.

Figure 3 displays two activities from Table 2 and provides an interpretation of the p-value of each segment. The red and blue line describe the distribution of these activities in segment 4 and 8, respectively. The pairwise comparison of activities in the left figure shows a similar monotonic behavior. On the other hand, the right figure shows different behavior for the two activities. Even though both start with similar behavior, the 2nd halves of behavior are quite different. Indeed the applied test correctly assigns a lower p-value to the right figure.

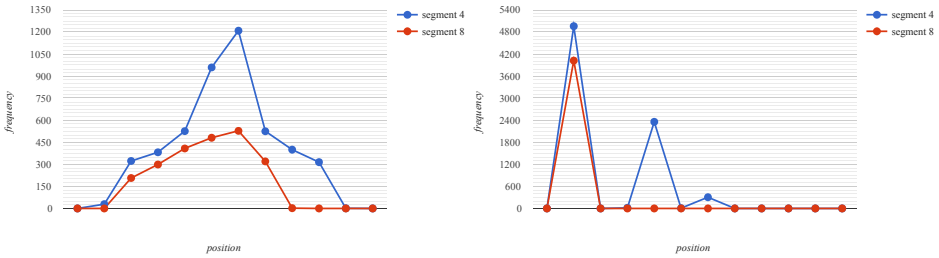

Fig. 3. Comparison between p-value of activities f(left) and b(right). In the picture on the left the probability distribution is monotonic, in the picture on the right it is not.

4 Case study

The method we described in the previous sections was applied to a real-life case study involving a manufacturing company in Italy. The event log collected by this company includes different business process related to product life-cycle management. Table 4 lists some descriptive statistics about this event log. The aim of the company was to discover real-life models that can be then used as a reference to identify cases that are deviating from the norm. In order to identify significant segments in the event log we considered Business Rules (BR) as criteria to construct segments. Each segment only includes cases consistent to a specific business rule. Then we used the Kendall's test as a metric for clustering segments.

Table 4. Descriptive Statistics of the event log

#events	**#cases**	**mean case duration**	**median case duration**	**min duration**	**max duration**
94622	24858	61 hrs	7 secs	0 mills	300 days

A comparison of the results is shown in Fig. 4 where a dendrogram, or tree diagram, is used to illustrate the hierarchical arrangement of the clusters obtained. The thick red line in the figure helps to cut the dendrogram and returns the group of samples that belongs to the same cluster. By adjusting the assessment criterion, we can have more or less detailed group of segments in each cluster.

The next step is to perform process discovery for each cluster, as shown in Fig. 5. Significance of discovered model has been tested by asking to three managers of this company to rate in a Likert scale their agreement with the following sentence "*Do you think the model discovered improves your understanding of this business process* ?". According to these managers, the models discovered using BR=(6,8) and BR=(9,10) are significant, corroborating with the discovered clusters [10]. On the other hand, the model discovered by BR=(1,2,3,4,5,7) did not improve their understanding of the business process, when compared to the model obtained from the entire event log [11]. Therefore, additional analysis is required for this cluster. Indeed, to fit the assessment criterion, quite a number of segments were included in this cluster, indicating that the applied segmentation was not really able to characterise a model.

5 Time Complexity Analysis

In this section we provide an evaluation of our method in terms of time complexity, and we compare it to other techniques available in the literature. As

[10] The rates provided are 3 "Neither agree nor disagree" , 4 "Agree", and 4 "Agree ".
[11] The rates provided are 1 "Strongly disagree" , 2 "Disagree", and 2 "Disagree ".

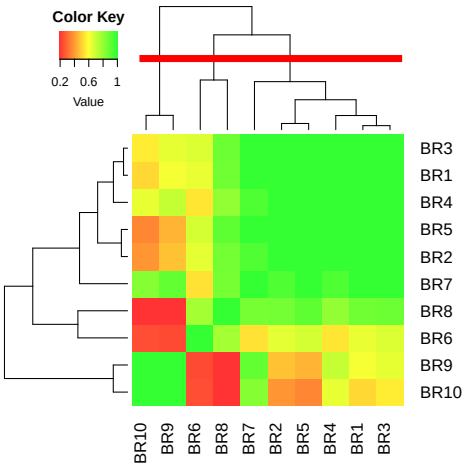

Fig. 4. Comparison table obtained with Kendall's Tau hypothesis test with p-value=0.05. The red line shows the cut line we applied for obtaining clusters compliant to the adopted assessment criteria. The clusters discovery were BR=(6,8), BR=(9,10) and BR=(1,2,3,4,5,7). Colours represent similarity values, as reported in the legend on top left size.

discussed in Sect. 2 naive solutions implies computational costs that are linear in the log size. In comparison to these solutions our approach has a higher complexity. Nevertheless, naive models do not account the trace structure [**?**] while our technique encode structural information in trace profiling. Indeed the time complexity we achieve is less than the one of other solutions taking into account the structure of the event log that, as already reported in Sect. 2, have to introduce some exponential factor.

In order to calculate the overall time complexity, we perform our analysis in three steps, so that it will be easier to understand. The evaluation of the proposed approach has focused on the contribution of statistical inference towards supporting the similarity of activities. In other words, the complexity of other involved techniques was not considered in our evaluation upon highlighting the main contribution. After identifying the Jensen-Shannon divergence, we calculate the complexity of Spearmans's rank, our non-parametric hypothesis test algorithm. Then we calculate the complexity of clustering method (Kendall's rank) and finally the final evaluation is given by higher complexity achieved.

5.1 Complexity of Jensen-Shannon divergence test

As mentioned in Sect. 3.3, the Jensen-Shannon calculation uses KL computation to obtain an initial segment distance. KL has $\mathcal{O}(a * p)$ where a is the number of activities and p is the possible positions acquired by activities. The other opera-

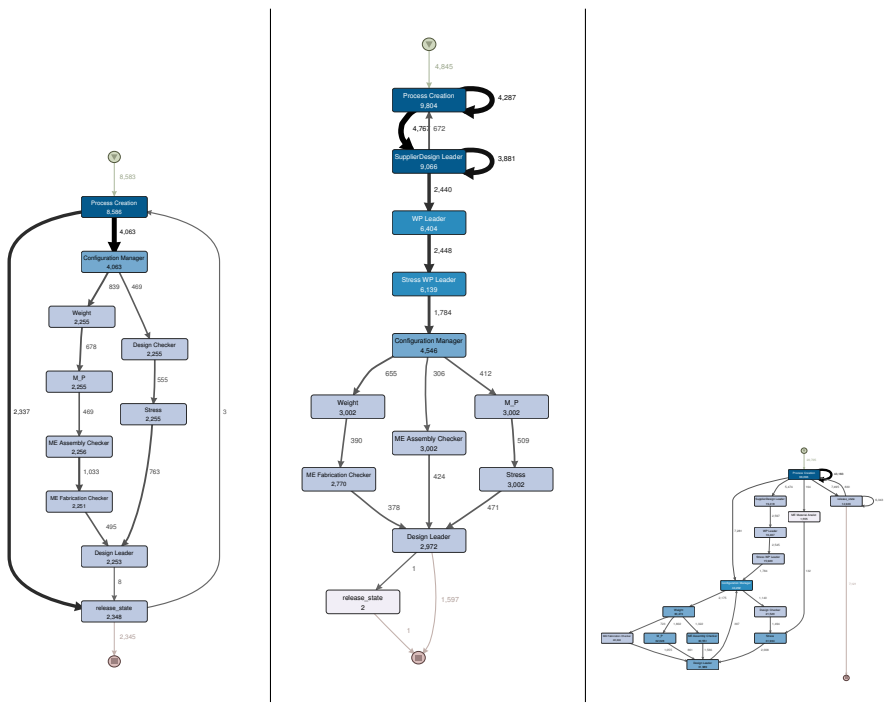

Fig. 5. From left to right, discovery model of BR=(6,8), BR=(9,10) and BR=(1,2,3,4,5,7).

tions included in the Jensen-Shannon test have constant asymptotic complexity. This way, the final complexity is $\mathcal{O}(a * p + 1) = \mathcal{O}(a * p)$.

5.2 Complexity of Spearman's rank correlation test

The complexity of Spearman's rank correlation test for two lists $x_1, \cdots, x_p$ and $y_1, \cdots, y_p$ is calculated as follows:

1. no tied ranks: $\rho = 1 - \left(\frac{6 \sum d^2}{p(p^2-1)}\right)$;
2. tied ranks: $\rho = \frac{\sum_i (x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_i (x_i - \overline{x})^2 \sum_i (y_i - \overline{y})^2}}$.

Where p is the maximum number of positions in a segment. The formula for no tied ranks has fewer operations than the tied rank formula; so we calculate only the complexity of the second formula. There are 2 averages, $2p$ differences, three sums with p summands and 1 division, 1 multiplication and 1 square root. Then the complexity will be $\mathcal{O}(2 + 2p + 3p + 1 + 1 + 1) = \mathcal{O}(p)$.

Before applying the formula, we need to sort the variables and obtain their ranks. Depending on the sorting algorithm, we can have different complexity. Best general sorting algorithms (such as Binary Tree Sort, Merge Sort, Heap Sort, Smooth Sort, Intro Sort, etc.) have the worst case complexity of

$\mathcal{O}(p\ log(p))$. The overall complexity of Spearman's rank correlation test is the sum of the above steps which is $\mathcal{O}(p\ log(p)) + \mathcal{O}(p) = \mathcal{O}(p\ log(p))$.

5.3 Complexity of Kendall's rank correlation test

In order to compute the number of concordance, discordance and ties, required to compute this test we need to compare each position with itself in a brute-force manner. If we consider all permutations of positions and eliminate comparison of position with itself, we obtain $\frac{p^2}{2} - p$ comparison which has the complexity of $\mathcal{O}(p^2)$.

In [**?**] were described sorting procedures that reduce this complexity. The basic idea is to sort the observation in one dimension and then sort this sorted values in the other dimension using a modified version of merge sort. This modified version takes advantage of having sorted values in the first dimension. As the complexity of merge sort of the algorithm in [**?**] is $\mathcal{O}(n\ log(n))$, the complexity of Kendall's rank correlation test can be reduced to $\mathcal{O}(p\ log(p))$.

6 Conclusion and Future Work

In this paper we described a novel method for improving the characterisation of event logs in preparation to PM. Our original contribution covers different aspects:

- We highlighted the different steps that must be integrated to work out a preprocessing task, underlining that a consistent representation of the different elements involved in these steps is required to support multiple iterations.
- We proposed a method for trace profiling tath brings to trace clustering with linearithmic time complexity. Comparing it with approaches with higher complexity or similar complexity that require treatment over all activities in the log, we claim that our method can improve process modelling without increase the complexity of the computing effort.
- We proposed the adoption of distance metrics rooted in inferential statistics supporting explicit assumptions on the probability distribution that are used in tests or to get specific characterisation about the correlation between two distributions.
- We applied the proposed methodology in a case study to demonstrate its positive applicability.

Future work will develop along several avenues. On the one hand, the method we adopted to generate position profile can be refined to get additional sensitivity to structural information. For example, it is of interest to verify how duplicated or skipped activities impact on similarity measures. In this way, we will experiment several clustering approaches and the impact of different distance metrics on their performances. Furthermore, additional statistical tests can be considered, in particular for supporting multi-vector comparison. Finally, the assessment procedure can be enriched introducing maximisation criteria insisting on inter- and intra-cluster distances.

Acknowledgements

This work was partly supported by the project "Cloud-based Business Process Analysis" funded by the Abu Dhabi ICT Fund at EBTIC/Khalifa University