A Generic Framework for Attribute-Driven Hierarchical Trace Clustering

Sebastiaan J. van Zelst[1,2] and Yukun Cao[2]

[1] Fraunhofer Institute for Applied Information Technology (FIT), Sankt Augustin, Germany
sebastiaan.van.zelst@fit.fraunhofer.de
[2] Chair of Process and Data Science, RWTH Aachen University, Aachen, Germany

Abstract. The execution of business processes often entails a specific process execution context, e.g. a customer, service or product. Often, the corresponding event data logs indicators of such an execution context, e.g., a customer type (bronze, silver, gold or platinum). Typically, variations in the execution of a process exist for the different execution context of a process. To gain a better understanding of the global process execution, it is interesting to study the behavioral (dis)similarity between different execution contexts of a process. However, in real business settings, the exact number of execution contexts might be too large to analyze manually. At the same time, current trace clustering techniques do not take process type information into account, i.e., they are solely behaviorally driven. Hence, in this paper, we present a hierarchical data-attribute-driven trace clustering framework that allows us to compare the behavior of different groups of traces. Our evaluation shows that the incorporation of data-attributes in trace clustering yields interesting novel process insights.

Keywords: Process mining · Trace clustering · Process comparison.

1 Introduction

Modern information systems employed at companies track the execution of business processes in great detail. The activities executed in the context of a business process, i.e., *events*, are stored in *event logs*. Automated analysis of event logs allows one to get a better understanding of the process based on what happened in reality (as recorded in the information system). In the field of *process mining* [1], several techniques have been developed that provide such automated analyses, e.g., methods exist that automatically construct a process model that describes the process as captured by the event data [2].

Ideally, process mining techniques allow us to get instant insights into the execution of a process, based on an event log. However, the direct application of an automated process discovery algorithm on a real event log often leads to a complex process model which is hard or even impossible to comprehend. This is typically caused by incorrect logging of events and low-frequent executions of the process that severely deviate from the process' main flow. Some authors have proposed methods to preprocessing/filter event data [10, 16–18], however, even properly filtered data often yields imprecise and complex process models. In some cases, this is because a company executes the process slightly different for the different *process execution contexts* of its end product and/or service. Even if the differences between the process execution are subtle, e.g., just swapping two activities, state-of-the-art process discovery algorithms tend to discover process models of inferior quality.

Process execution contexts are omnipresent, yet, little to no work focuses on providing end-to-end solutions to exploit event data describing execution contexts. To some degree, e.g., when a company distinguishes between 4 different customer types, manually slicing the data into relevant subsets and subsequently analyzing/comparing the corresponding process models is feasible. However, often, the number of process categories is too large to perform such an analysis manually. For example, consider the WABO/CoSeLoG event data [8], which describes different sub-logs capturing how five Dutch municipalities handle the application of building permits. In its current form, a comparative study is still feasible, however, in case one collects data of the same process among all Dutch municipalities (>300), such a manual analysis is no longer feasible.

Therefore, in this paper, we propose a framework that allows for attribute-driven hierarchical clustering of traces. Given a user-defined case attribute, we construct a hierarchical clustering of the given event data, grouping the most similar executions of the process. As such, the framework allows the user to inspect collections of similar process executions belonging to different process contexts. The clusters allow us to group different process contexts that have a similar process execution. Furthermore, the clusters allow us to improve the overall process performance of specific process execution contexts based on similar process contexts. The accompanying implementation, built on top of the PM4Py framework [4], allows the user to compare different groups in the hierarchy. Using the implementation, we conducted a set of large-scale experiments based on publicly available real event data sets. Our experiments consistently show that the models found based on attribute-driven event logs are of superior quality compared to the models discovered by process discovery algorithms when directly applied on the original event data.

The remainder of this paper is structured as follows. In Section 2, we present background concepts. In Section 3, we present the proposed framework. In Section 4, we present the results of the evaluation conducted in the context of this paper. In Section 5, we discuss related work. Section 6, concludes this work.

2 Background

In this section, we conceptually present *event logs* and *hierarchical clustering.*

Event Logs Event logs are the primary source of data for almost any process mining analysis. An event log captures at which point in time an activity was ex-

Table 1: Example event log (a), adopted from [1], describing a compensation request process for concert tickets and (b) an exemplary mapping of case identifiers to data attributes, i.e., each case id has an associated ticket class.

(a)

Case id	*Event id*	*Activity*	*Timestamp*	*Resource*	⋯
⋮	⋮	⋮	⋮	⋮	⋯
1273	4632	register request	12-11-2019 11.02	Barbara	⋯
1274	4633	register request	12-11-2019 11.32	Jan	⋯
1273	4634	check ticket	12-11-2019 12.12	Stefanie	⋯
1274	4635	examine casually	12-11-2019 14.16	Jorge	⋯
1275	4636	register request	12-11-2019 14.32	Josep	⋯
1275	4637	examine thoroughly	12-11-2019 15.42	Marlon	⋯
1273	4638	examine thoroughly	13-11-2019 11.18	Barbara	⋯
1273	4639	decide	13-11-2019 15.34	Wil	⋯
1273	4640	reject request	13-11-2019 16.50	Arthur	⋯
⋮	⋮	⋮	⋮	⋮	⋯

(b)

Case id	*Ticket type*
⋮	⋮
1273	VIP
1274	Basic
1275	Basic
1276	Plus
1277	Royal
⋮	⋮

ecuted in the context of an instance of a process. Consider Table 1a, in which we present a simplified example of an event log. The first column depicts the *Case identifier*, i.e., identifying a specific instance of a process, e.g., a concert ticket. The *Activity* column registers what activity was performed for the corresponding case. The *Timestamp* column registers at what time the activity occurred. Observe that other data attributes, related to the activities, are available in an event log as well, e.g., the *Resource* and *Cost* columns in Table 1a.

For each process instance, multiple activities are logged over time, e.g., for the process instance with case-id 1273 we observe the sequence ⟨register request, check ticket, ..., reject request⟩. The execution of an activity in the context of a process instance is referred to as an *event* (identified by the *Event id* column). Observe that events carry auxiliary data payload, e.g., the *Resource* column. Typically, process instances (identified by the case identifier) also carry data attributes. Consider Table 1b, in which we depict an example of such a *case attribute*, corresponding to the event log in Table 1a, i.e., where each compensation request (Case id) is related to a ticket class. In general, different case attributes may exist, e.g., in what country/factory a product is produced, the ticket price, etc. Typically, a business owner is interested in assessing or comparing the execution of the process along the lines of such data attributes, e.g., in which branch of the company is the process executed most efficiently?

In the context of this paper, we assume an event log $L \subseteq \mathcal{C}$ to be a collection of cases ($\mathcal{C}$ denotes the universe of cases). Furthermore, given some $c \in \mathcal{C}$ data attribute of interest $d \in \mathcal{D}$ ($\mathcal{D}$ denotes the universe of data attributes), we assume that we are able to retrieve the value of attribute d by means of $\pi_d(c)$. In particular, for $\texttt{trace} \in \mathcal{D}$, we assume that $\pi_{\texttt{trace}}(c) \in \mathcal{A}^*$ (where $\mathcal{A}^*$ denote the set of all sequences over the activity universe $\mathcal{A}$). Finally, we let $\hat{\mathcal{D}} = \mathcal{D} \setminus \{\texttt{trace}\}$ and we overload the notation for attribute projection (π) to sets of cases, i.e., given $L \subseteq \mathcal{C}$, we have $\pi_d(L) = \{\pi_d(c) \mid c \in L\}$.

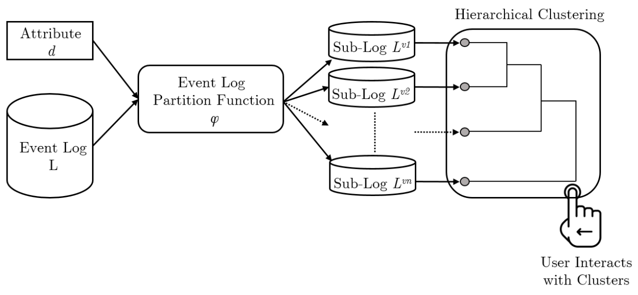


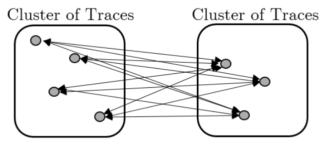
Fig. 1: Overview of the proposed framework. The event log is partitioned based on a user-defined data attribute $d \in \mathcal{D}$, i.e., by means of partition function φ. The corresponding resulting event logs, i.e., $L^{v_1}, L^{v_2}, ..., L^{v_n}$, are combined into a hierarchical clustering based on the behavior they describe.

Hierarchical Clustering The framework proposed in this paper builds a hierarchical clustering of cases. A detailed description of hierarchical clustering is out of scope, i.e., we refer to [27] for an elaborate overview. Informally, a hierarchical clustering of a set of (sets of) elements defines a hierarchy of clusters based on these aforementioned (sets of) elements. Mathematically, a hierarchical clustering is a binary tree of some height h. The bottom layer of the hierarchy, i.e., the nodes at height 0, is defined by the individual elements of the set (or an initial clustering). The top layer of the hierarchy, i.e., the root at height h, is the complete set of elements. A cluster C_i at height $0<i<h$ connects to, i.e., is a super-set of, at least one cluster C_{i-1} at height $i-1$ and a cluster C_j at height $0 \leq j < i$. Given a distance measure on the clusters, a cluster combining two other clusters implies that the distance among these two clusters is minimal.

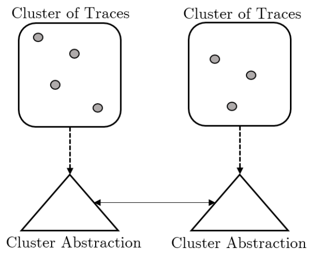
3 Attribute-Driven Hierarchical Trace Clustering

Consider Fig. 1, in which we depict a schematic overview of the proposed framework for data-driven hierarchical clustering. Given *attribute* $d \in \mathcal{D}$ and event log $L \subseteq \mathcal{C}$, we partition the cases in the event log (using partition function φ). The partitioning groups cases together that have the same value for data attribute $d \in \mathcal{D}$. Hence, given that $|\pi_d(L)|=n$, the partitioning function φ yields n sub-logs ($L^{v_1}, ..., L^{v_n}$ in Fig. 1) that form a partition of the original event log. By definition, $|\pi_d(L')|=1$, $\forall L' \in \varphi(L, d)$, i.e., each sub-log defined by the partition is uniquely associated to one value $v \in \pi_d(L)$.

Given the initial partitioning as defined by $\varphi(L, d)$, we construct a *hierarchical clustering*, using the initial partitioning as a primary input, i.e., the hierarchy's nodes at height 0. To construct a hierarchy of clusters, we require a distance function, i.e., a *linkage criterion*, that allows us to compute the distance between the hierarchy clusters, i.e., the sub-logs. Formally, we require a linkage criterion of the form $\Delta\colon \mathcal{P}(\mathcal{C}) \times \mathcal{P}(\mathcal{C}) \rightarrow \mathbb{R}_{\geq 0}$. However, we are primarily interested in the



(a) *Trace-based linkage criterion computation*; Cluster distances are aggregations of the inter-trace distance of the elements of the different clusters.



(b) *Abstraction-based linkage criterion computation*; Inter-cluster distance is computed in terms of a distance based on abstractions of the cluster.

Fig. 2: Overview of the two different ways to compute linkage criteria considered.

linkage criterion in terms of the *control-flow behavior* of the cases, i.e., in terms of $\pi_{\texttt{trace}}(c){\in}\mathcal{A}^*$. Using $\pi_{\texttt{trace}}(c){\in}\mathcal{A}^*$, $\forall c{\in}\mathcal{C}$, we consider two forms of linkage criterion computation, i.e., *trace-based linkage computation* and *abstraction-based linkage computation*. Consider Fig. 2, in which we schematically depict the two different linkage computation strategies.

In *trace-based linkage computation*, the linkage criterion Δ is computed by means of computing some aggregate over a trace-based distance function $\delta\colon \mathcal{A}^*\times\mathcal{A}^*{\to}\mathbb{R}_{\geq 0}$. Observe that a wide variety of such aggregates exists, e.g., *maximal linkage clustering* with, given $X,Y{\subseteq}\mathcal{C}$, $\Delta(X,Y){=}\max\{\Delta(\pi_{\texttt{trace}}(c),\pi_{\texttt{trace}}(c'))\mid c{\in}X, c'{\in}Y\}$, etc. In *abstraction-based linkage computation*, we compute an abstraction of the behavior described by the collection of cases, which we subsequently use to compute the distance measure, e.g., by first translating the traces of each cluster into a prefix-tree and computing a distance function over the corresponding prefix trees.

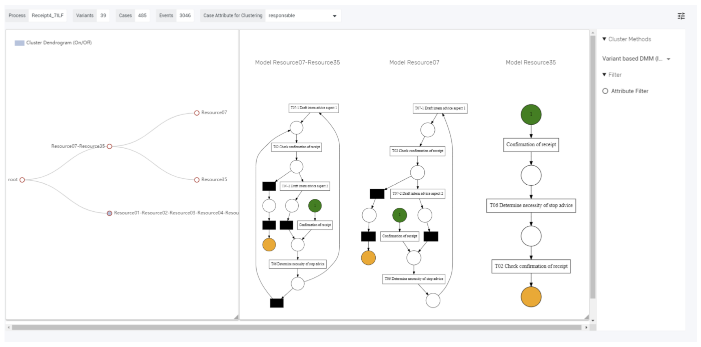
4 Evaluation

Here, we evaluate the application of the framework. In Section 4.1, we present the implementation. In Section 4.2, we present the results of the evaluation.

4.1 Implementation

The proposed framework is implemented on top of the PM4Py [4] process mining library and is publicly available.[1] In the prototype (Fig. 3), the user can select which attribute to use as a main driver for the initial partitioning. In the left-hand side of the application, the hierarchical clustering is shown. The user can

[1] https://github.com/caoyukun0430/pm4py-source/tree/yukun_paper and https://github.com/caoyukun0430/pm4py-ws/tree/dev-yukun



Process Receipt4_7ILF | Variants 39 | Cases 485 | Events 3046 | Case Attribute for Clustering responsible

Cluster Methods: Variant based DMM (I...

Cluster Dendrogram (On/Off)
root
Resource07-Resource35
Resource07
Resource35
Resource01-Resource02-Resource03-Resource04-Resou
Model Resource07-Resource35
Model Resource07
Model Resource35
T07-1 Draft intern advice aspect 1
T02 Check confirmation of receipt
T07-2 Draft intern advice aspect 2
Confirmation of receipt
T06 Determine necessity of stop advice
Filter
Attribute Filter

Fig. 3: Screenshot of an interactive prototype of the clustering framework.

select different nodes of the clustering, to inspect the corresponding discovered process model, i.e., based on the event data belonging to the cluster. Different distance/linkage functions can be selected from the right-hand-side menu.

We used the implementation to conduct our experiments. In the implementation, several instantiations of the required distance/linkage functions, i.e., inter-trace distance function δ, linkage criteria and abstraction-based distances are available. We briefly describe these instantiations here.

Inter-Case Distance Metrics Two different inter-case measures, i.e., instantiations of the δ-function, are available, i.e., *Levenshtein distance* and *Behavioral-Trace Distance*:

- *Levenshtein distance* [24](δ_L); Expresses the amount of edits (insertion, removal or replacements) we need, in order to transform a given sequence σ into another sequence σ'. The Levenshtein distance between $\langle a,b,c\rangle$ and itself is 0, whereas the Levenshtein distance between $\langle a,b,c\rangle$ and $\langle a,c\rangle$ is 1.
- *Behavioral-Trace Distance* (δ_B^{α}); Transforms two given traces into behavioral abstraction vectors, on which we compute a vector based distance. We consider two different behavioral vectors:
  - *Activity occurrence abstraction*, i.e., given $\sigma{\in}\mathcal{A}^*$, we define a vector $\vec{\sigma}_a{\in}\mathbb{N}^{\mathcal{A}}$, where $\vec{\sigma}_a(a){=}|\,\{i{\in}\{1,...,|\sigma|\}\mid\sigma(i){=}a\}\,|$.
  - *Subsequent relation occurrence abstraction*, i.e., given $\sigma{\in}\mathcal{A}^*$, we define a vector $\vec{\sigma}_s{\in}\mathbb{N}^{\mathcal{A}\times\mathcal{A}}$, where $\vec{\sigma}_s(a,a'){=}|\{i{\in}\{1,...,|\sigma|-1\}\mid\sigma(i){=}a{\wedge}\sigma(i+1){=}a'\}|$.

  Given the two aforementioned abstractions, and, parameter $\alpha{\in}[0,1]{\subseteq}\mathbb{R}$, we define the behaviorally driven inter-trace distance function as:

$$\delta_B^{\alpha}(\sigma,\sigma'){=}\alpha\left(1-\frac{\vec{\sigma}_a{\cdot}\vec{\sigma}'_a}{||\vec{\sigma}_a||\ ||\vec{\sigma}'_a||}\right)+(1-\alpha)\left(1-\frac{\vec{\sigma}_s{\cdot}\vec{\sigma}'_s}{||\vec{\sigma}_s||\ ||\vec{\sigma}'_s||}\right) \qquad (1)$$

Trace-Based Linkage Criteria Next to the two trace-based distance metrics, we implemented two trace-based linkage criteria, i.e., *unweighted average linkage clustering (UPGMA)* and *dual minimal match linkage criterion (DMM)*:

- *Unweighted Average Linkage Clustering (UPGMA)*; Given two clusters of cases, i.e., $L, L' \subseteq \mathcal{C}$, we compute:

$$\frac{1}{|L|\ |L'|} \sum_{c \in L} \sum_{c' \in L'} \delta\left(\pi_{\texttt{trace}}(c), \pi_{\texttt{trace}}(c')\right) \tag{2}$$

- *Dual Minimal Match (DMM)*; Given two clusters of cases, i.e., $L, L' \subseteq \mathcal{C}$, we compute:

$$\frac{1}{|L|+|L'|}\left(\sum_{c \in L} \min\left\{\delta\left(\pi_{\texttt{trace}}(c), \pi_{\texttt{trace}}(c')\right) \mid c' \in L'\right\} + \sum_{c' \in L'} \min\left\{\delta\left(\pi_{\texttt{trace}}(c), \pi_{\texttt{trace}}(c')\right) \mid c \in L\right\}\right) \tag{3}$$

Abstraction-Based Linkage Criteria In the implementation, we consider the same abstraction as used in the inter-trace behavioral distance metric, i.e., behavioral abstraction vectors capturing activity occurrence and subsequent relations. However, we first aggregate the behavioral relations, i.e., based on the members of a cluster, prior to computing the behavioral vectors. or convenience, we formalize the two abstractions on the cluster-level.[2]

- *Activity occurrence abstraction*, i.e., given $C \in \mathcal{P}(\mathcal{A}^*)$, we define a vector $\vec{C_a} \in \mathbb{N}^{\mathcal{A}}$, where $\vec{C_a}(a) = \sum_{\sigma \in C} |\{i \in \{1, ..., |\sigma|\} \mid \sigma(i) = a\}|$.
- *Subsequent relation occurrence abstraction*, i.e., given $C \in \mathcal{P}(\mathcal{A}^*)$, we define a vector $\vec{C_s} \in \mathbb{N}^{\mathcal{A} \times \mathcal{A}}$, where $\vec{C_s}(a, a') = \sum_{\sigma \in C} |\{i \in \{1, ..., |\sigma| - 1\} \mid \sigma(i) = a \wedge \sigma(i+1) = a'\}|$.

Given the cluster vectors, we are able to compute the linkage criterion using the weighted cosine similarity-based distance metric, presented in Equation 1.

In total, we consider 5 different instantiations of the framework, i.e., trace-based instantiations δ_L-UPGMA, δ_L-DMM, δ_B^{α}-UPGMA and δ_B^{α}-DMM and the vector-based abstraction-based linkage V-ABL.

4.2 Results

Here, we present the results of the evaluation of applying attribute-driven hierarchical clustering. We briefly present qualitative results on real event data, after which we quantitatively assess the approach on several real event logs.

Qualitative Evaluation on Real Event Data In this section, we present the qualitative clustering results of our framework applied to real event data. We use the *Business Process Intelligence Challenge (BPIC)* 2017 Offer [14] event data, which describes all the offers made for a *loan application processes.* We select the *CreditScore* attribute containing 520 different values, ranging from 0 to 1145, since we are interested in discovering typical application behaviors on customers under different credit scores.

[2] We formalize the abstractions on sets of sequences over $\mathcal{A}$, rather than elements $c \in \mathcal{C}$. Note that, given $c \in \mathcal{C}$, we are able to access the trace-view by means of $\pi_{\texttt{trace}}(c)$.

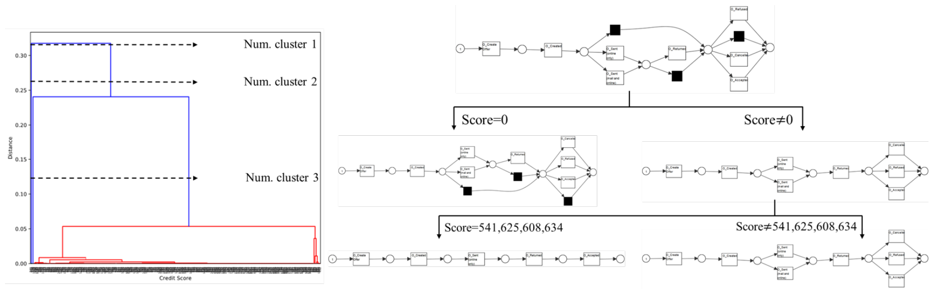
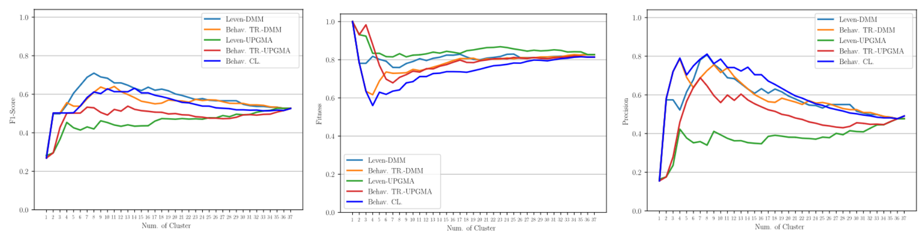


Fig. 4: Truncated dendrogram at the top three hierarchy levels with the corresponding discovered process models ((BPIC) 2017 Offer [14] event data).

As depicted in Fig. 4, the algorithm separates the data into clusters with a *zero (*0*)* and a *non-zero (*≠0*) Score* value. Corresponding event log sizes are 27735 and 15260 respectively. The underlying discovered models are relatively similar (depicted on the right-hand side of Fig. 4), i.e., for Score values equal to 0, more skipping of activities is allowed. When we dive deeper into the data, we observe that the *O_Cancelled* and the *O_Refused* activity dominate the sub-log under the zero value (total occurrence 80.35%). At the same time, the *O_Accepted* activity dominates the traces under non-zero values (94.38%). This difference leads to the vast behavioral inter-cluster distances and explains the clustering found. *In particular, this matches the expectation that people without a credit score get rejected more often when applying for loans in reality.* As we traverse the hierarchy further, we observe one outlier cluster containing only four cases with credit scores "541, 608, 625, 634", in which only the single behavior *O_Sent (online only)* is allowed. Compared to this behavioral cluster, the remaining cluster excluding these four values shows a large variety of behavior. Therefore, the clustering dendrogram in Fig. 4 provides us a clear understanding of different customer behaviors under different credit scores as well as extracting outlier behavior from the complete model.

Quantitative Evaluation on Real Event Data In this section, we show the results of applying our implementation on a variety of different publicly available event logs. We first assess the impact of the different distance metrics and linkage criteria based on the "Receipt phase of an environmental permit application process ('WABO'), CoSeLoG project" event log ("Receipt log") [7]. Additionally, we assess how the technique performs on several different publicly available event logs.

In Fig. 5, we present average $F1$-, replay-fitness- and precision scores, obtained for the models discovered based on the Receipt log, for the five different implemented distance metrics and linkage criteria. Replay-fitness (range $[0,1] \subseteq \mathbb{R}$) indicates how well a model describes a given event log (value 1 implies that the model describes all behavior in the event log). Precision (range $[0,1] \subseteq \mathbb{R}$) quantifies the amount of additional behavior described by the model (value 1

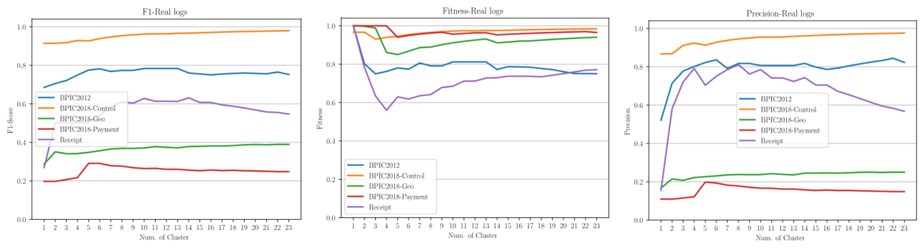


(a) F1 Score (b) Replay-Fitness Score (c) Precision Score

Fig. 5: Average F1, replay-fitness and precision scores obtained for the models discovered based on the Receipt log [7], for the different distance metrics and linkage criteria as described in this paper.

implies that all described behavior is part of the input event log). The F1 score is the harmonic mean of replay-fitness and precision. We used the `responsible` attribute as a basis for the hierarchy, and, for abstraction based linkage, we use $\alpha{=}\frac{1}{2}$. The x-axis of the charts describes the number of separate clusters in the hierarchy, i.e., starting in the root of the hierarchy and traversing down. We observe an increase in the F1 score, which gradually decreases, yet, remains above the F1 score obtained for the model discovered based on the whole event log (the hierarchy's root). When going deeper into the hierarchy, the increase in the F1 score is explained by a higher increase in precision when compared to the corresponding decrease in replay-fitness. However, when traversing the hierarchy further, surprisingly, the replay-fitness slightly increases whereas the precision slightly drops. Upon inspection of the results, we observe that the higher levels of the hierarchy contain outlier clusters that have a relatively high corresponding precision value and relatively low replay-fitness value. However, deeper in the hierarchy, more similar clusters are "split-up" into the lower levels of the hierarchy. In some cases, the newly added fitness values in a deeper level in the hierarchy are higher than the average fitness values of the current level of the hierarchy, leading to a higher average fitness value in the next layer. The same holds, symmetrically, for the corresponding precision values, explaining the results in Fig. 5.

In Fig. 6, we present the results F1, replay-fitness and precision scores obtained for the models discovered based on different publicly available event logs, using V-ABL. Observe that, in the plots provided here, we only plot the highest 23 levels of the hierarchy (in most cases the hierarchies tend to be higher). The event logs considered are BPIC 2012 [13], three different sub-logs (Control, Geo and Payment) of BPIC 2018 [15] and the Receipt log (also used in the previous section). Selection of the trace attributes is based on manual inspection of the results. We observe similar results on the different logs, i.e., slight increases of the F1 scores due to increases in precision and smaller decrease (and stabilizing) fitness values. Again, stabilization of the fitness values is due to the initial identification of outlier clusters.



(a) F1 Score (b) Replay-Fitness Score (c) Precision Score

Fig. 6: Average F1, replay-fitness and precision scores obtained for the models discovered based on different publicly available event logs, based on abstraction-based linkage criterion. In all cases, we observe a slight increase in the F1 score, primarily driven by increases in precision.

5 Related Work

An overview of the field of process mining is out of the scope of this paper, i.e., we refer to [1] for an introduction to the field. In the remainder, we explicitly focus on related work in the area of *trace clustering*. In particular, we briefly provide a generalized overview of different existing clustering techniques, i.e., in general, the majority of the different clustering techniques proposed can be integrated into the framework presented in this paper.

Several authors focus on trace clustering based on a *feature vector based distance* [3,6,11,12,20,22,23,25,26,28,29]. These techniques consider different ways to extract specific features from the activity traces, which are subsequently translated into a feature vector. Examples of such features are event frequency, direct succession frequency, etc. For a given feature vector, different linkage criteria and clustering algorithms can be used to compute the final clustering. Alternatively the *syntactic distance* between traces, e.g., Levensthein, is used [5]. Another commonly used type of clustering focuses directly on *models*, rather than individual traces [9,11,19,21,25,30,31]. The goal of these methods here is to obtain a better model (i.e., higher precision, fitness) by merging traces into clusters. Typically traces are added into a cluster if they improve the quality of the cluster-based process model.

6 Conclusion

Processes executed in companies, hospitals, etc. are often performed in the context of an *execution context*, e.g., a customer type. Often, slight variations exist in the execution of the process for the different execution contexts. Manual comparison of the differences in execution, based on event data stored in event logs, is no longer feasible for more significant numbers of execution contexts. Hence, in this paper, we presented a hierarchical trace clustering framework that allows us to perform behavioral trace clustering over groups of traces, that share common data attributes. We evaluated the approach using several real data sets, based on several different behavioral comparison techniques. Our evaluation shows that the models described by the different clusters of the hierarchy are of better quality compared to the models discovered on the event data as a whole.

References

1. van der Aalst, W.M.P.: Process Mining - Data Science in Action, Second Edition. Springer (2016)
2. Augusto, A., Conforti, R., Dumas, M., Rosa, M.L., Maggi, F.M., Marrella, A., Mecella, M., Soo, A.: Automated discovery of process models from event logs: Review and benchmark. IEEE Trans. Knowl. Data Eng. **31**(4), 686–705 (2019)
3. Bae, J., Caverlee, J., Liu, L., Yan, H.: Process mining by measuring process block similarity. In: International Conference on Business Process Management. pp. 141–152. Springer (2006)
4. Berti, A., van Zelst, S.J., van der Aalst, W.M.P.: Process mining for python (PM4Py): Bridging the gap between process-and data science. In: Proceedings of the ICPM Demo Track 2019, co-located with 1st International Conference on Process Mining (ICPM 2019), Aachen, Germany, June 24-26, 2019. p. 13–16 (2019)
5. Bose, R.J.C., Van der Aalst, W.M.: Context aware trace clustering: Towards improving process mining results. In: Proceedings of the 2009 SIAM International Conference on Data Mining. pp. 401–412. SIAM (2009)
6. Bose, R.J.C., van der Aalst, W.M.: Trace clustering based on conserved patterns: Towards achieving better process models. In: International Conference on Business Process Management. pp. 170–181. Springer (2009)
7. Buijs, J.: Receipt phase of an environmental permit application process ('wabo'), coselog project (2014). https://doi.org/10.4121/UUID:A07386A5-7BE3-4367-9535-70BC9E77DBE6
8. Buijs, J.: Environmental permit application process ('WABO'), CoSeLoG project. Eindhoven University of Technology. Dataset. (2014). https://doi.org/10.4121/uuid:26aba40d-8b2d-435b-b5af-6d4bfbd7a270
9. Cadez, I., Heckerman, D., Meek, C., Smyth, P., White, S.: Model-based clustering and visualization of navigation patterns on a web site. Data mining and knowledge discovery **7**(4), 399–424 (2003)
10. Conforti, R., La Rosa, M., ter Hofstede, A.H.M.: Filtering out infrequent behavior from business process event logs. IEEE Trans. Knowl. Data Eng. **29**(2), 300–314 (2017)
11. De Koninck, P., Nelissen, K., Baesens, B., vanden Broucke, S., Snoeck, M., De Weerdt, J.: An approach for incorporating expert knowledge in trace clustering. In: International Conference on Advanced Information Systems Engineering. pp. 561–576. Springer (2017)
12. De Medeiros, A.K.A., Guzzo, A., Greco, G., Van Der Aalst, W.M., Weijters, A., Van Dongen, B.F., Saccà, D.: Process mining based on clustering: A quest for precision. In: International Conference on Business Process Management. pp. 17–29. Springer (2007)
13. van Dongen, B.F.: Bpi challenge 2012 (2012). https://doi.org/10.4121/UUID:3926DB30-F712-4394-AEBC-75976070E91F
14. van Dongen, B.F.: Bpi challenge 2017 (2017). https://doi.org/10.4121/UUID:5F3067DF-F10B-45DA-B98B-86AE4C7A310B

15. van Dongen, B.F., Borchert, F.: Bpi challenge 2018 (2018). https://doi.org/10.4121/uuid:3301445f-95e8-4ff0-98a4-901f1f204972
16. Fani Sani, M., van Zelst, S.J., van der Aalst, W.M.P.: Improving process discovery results by filtering outliers using conditional behavioural probabilities. In: Business Process Management Workshops - BPM 2017 Barcelona, Spain, September 10-11, 2017, Revised Papers. pp. 216–229 (2017)
17. Fani Sani, M., van Zelst, S.J., van der Aalst, W.M.P.: Applying sequence mining for outlier detection in process mining. In: CoopIS, C&TC, and ODBASE 2018, Valletta, Malta, October 22-26, 2018, Proceedings, Part II. pp. 98–116 (2018)
18. Fani Sani, M., van Zelst, S.J., van der Aalst, W.M.P.: Repairing outlier behaviour in event logs. In: Business Information Systems - 21st International Conference, BIS 2018, Berlin, Germany, July 18-20, 2018, Proceedings. pp. 115–131 (2018)
19. Ferreira, D., Zacarias, M., Malheiros, M., Ferreira, P.: Approaching process mining with sequence clustering: Experiments and findings. In: International Conference on Business Process Management. pp. 360–374. Springer (2007)
20. Greco, G., Guzzo, A., Pontieri, L., Sacca, D.: Discovering expressive process models by clustering log traces. IEEE Transactions on Knowledge and Data Engineering **18**(8), 1010–1027 (2006)
21. Hompes, B., Buijs, J., Van der Aalst, W., Dixit, P., Buurman, J.: Discovering deviating cases and process variants using trace clustering. In: 27th Benelux Conference on Artificial Intelligence (BNAIC), November. pp. 5–6 (2015)
22. Jung, J.Y., Bae, J.: Workflow clustering method based on process similarity. In: International Conference on Computational Science and Its Applications. pp. 379–389. Springer (2006)
23. Jung, J.Y., Bae, J., Liu, L.: Hierarchical clustering of business process models. International Journal of Innovative Computing, Information and Control **5**(12), 1349–4198 (2009)
24. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet physics doklady. vol. 10, pp. 707–710 (1966)
25. Lu, X., Tabatabaei, S.A., Hoogendoorn, M., Reijers, H.A.: Trace clustering on very large event data in healthcare using frequent sequence patterns. In: International Conference on Business Process Management. pp. 198–215. Springer (2019)
26. Luengo, D., Sepúlveda, M.: Applying clustering in process mining to find different versions of a business process that changes over time. In: International Conference on Business Process Management. pp. 153–158. Springer (2011)
27. Murtagh, F., Contreras, P.: Algorithms for hierarchical clustering: an overview, II. Wiley Interdiscip. Rev. Data Min. Knowl. Discov. **7**(6) (2017)
28. Song, M., Günther, C.W., Van der Aalst, W.M.: Trace clustering in process mining. In: International Conference on Business Process Management. pp. 109–120. Springer (2008)
29. Song, M., Yang, H., Siadat, S.H., Pechenizkiy, M.: A comparative study of dimensionality reduction techniques to enhance trace clustering performances. Expert Systems with Applications **40**(9), 3722–3737 (2013)
30. Sun, Y., Bauer, B.: A novel top-down approach for clustering traces. In: International Conference on Advanced Information Systems Engineering. pp. 331–345. Springer (2015)
31. De Weerdt, J., 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)