





# Finding Structure in the Unstructured: Hybrid Feature Set Clustering for Process Discovery

Alexander Seeliger<sup>(✉)</sup>, Timo Nolle<sup></sup>, and Max Mühlhäuser

Telecooperation Lab, Technische Universität Darmstadt, Darmstadt, Germany  
{seeliger,nolle,max}@tk.tu-darmstadt.de

**Abstract.** Process discovery is widely used in business process intelligence to reconstruct process models from event logs recorded by information systems. With the increase of complexity and flexibility of processes, it is getting more and more challenging for discovery algorithms to generate accurate and comprehensive models. Trace clustering aims to overcome this issue by splitting event logs into smaller behavioral similar sub-logs. From these sub-logs more accurate and comprehensive process models can be reconstructed. In this paper, we propose a novel clustering approach that uses frequent itemset mining on the case attributes to also reveal relationships on the data perspective. Our approach includes this additional knowledge as well as optimizes the fitness of the underlying process models of each cluster to generate accurate clustering results. We compare our method with six other clustering methods and evaluate our approach using synthetic and real-life event logs.

**Keywords:** Knowledge discovery · Process discovery  
Trace clustering · Process mining · Business process intelligence

## 1 Introduction

Business process intelligence supports organizations to optimize and improve their business processes. In particular, *process mining* [1] helps to understand the actual use of information systems in various environments. The basis for process mining are event logs, recorded by process-aware information systems (PAISs), industrial machines or sensors. Event logs reflect the activities performed by employees or machines, allowing the analysis of the relationships between activities. Additionally, the event log may also store much more information, such as the executor of an activity and the context of the case, such as the vendor, used material or the customer.

An essential part of process mining is *process discovery* which is an unsupervised method for reconstructing a process model from an event log. The challenge is to find a model that accurately matches the recorded observations, but is also human interpretable. Many business processes in the real world are

often executed in highly flexible environments, such as health care or product development. Here a dense distribution of cases with a high variety of complex behavior can be found. In such scenarios, process discovery often produces a spaghetti-like model which suffers from inaccuracy and high complexity. Furthermore, behavior on other process perspectives (e.g., data attributes) is not considered which may also be interesting for process analysts. For example, in an hospital the same admission process may be applied to emergency and non-emergency patients. Although the sequence of activities may be the same for both kinds of patients, the underlying process may be different with respect to resource assignments or activity durations.

*Trace clustering* tries to overcome these issues by splitting the different observed behaviors into multiple sub-logs of similar behavior. For each sub-log, process discovery is then applied separately to retrieve more accurate and human interpretable process models. However, existing methods either only rely on the control-flow perspective or ignore the quality of the discovered models. In particular, including different process perspectives and providing accurate process models is challenging.

In this paper, we introduce a new clustering method that uses a hybrid feature set to consider multiple process perspectives and optimizes the fitness of the underlying process models. Inspired by our previous work [14] in which we clustered documents together into meaningful groups by combining the document content and the user behavior, we transfer the idea to trace clustering in process mining. Our method considers the control-flow and the data perspective to extract process behaviors on both views to split the event log into multiple sub-logs. The basic idea is that different process behaviors often depend on the context of the case, e.g., the product category or the customer. These differences may only be small on the trace level, but large on the data perspective. Our clustering approach additionally uses the case attributes to distinguish between the different process behaviors. We extract frequent itemsets using frequent pattern mining [10] to find common relationships between case attributes and use them for clustering. Studies have shown that approaches relying solely on the control-flow are unable to identify different process behaviors adequately [18]. Furthermore, our method automatically optimizes the fitness of the sub-log process models, ensuring that the underlying models sufficiently represent the clustered traces.

In summary, the contributions of this paper are as follows:

1. We provide a hybrid feature set clustering approach that splits event logs into sub-logs containing similar cases based on multiple perspectives.
2. Our approach automatically adapts to the given event logs and determines the optimal clustering parameters by applying the particle swarm optimization algorithm to optimize model fitness.
3. We provide a comprehensive evaluation of six other trace clustering methods in the domain of business process intelligence.

The paper is structured as follows. First, we introduce related work. Second, we introduce our method that combines the data and the control-flow perspective

to generate clusters. Third, we evaluate our method and discuss the results. Finally, we conclude with a discussion and a short summary.

## 2 Related Work

Our work is related to trace clustering in process mining, aiming to improve accuracy and interpretability of reconstructed process models by separating different behaviors on different process perspectives into multiple sub-logs. A summary and an evaluation framework of trace clustering methods are provided in [18]. The authors elaborate a systematic empirical analysis of different techniques and evaluate their applicability in two scenarios: the identification of different processes and the improvement of the understandability of the mined models. We classify the related work into *distance-based* and *model-based* trace clustering.

Distance-based trace clustering such as [4, 9, 16] use a vector space model on the event traces to segment the event log into smaller sub-logs. Greco et al. [9] use significant subsequences of activities and activity transitions to generate clusters of traces. Similar, Bose et al. [4] propose the use of different sequence similarity measures to find traces with similar behavior with respect to the order of activities. Alignment-based approaches can also be used for specifying the difference between traces, as for example presented in [7]. In [16] the use of log profiles is proposed, allowing to incorporate different perspectives into the clustering. Each profile describes its own vector space which can be separately used for clustering. A co-training strategy for multiple view clustering was presented by Appice et al. [2]. The authors combine multiple log profiles using unsupervised co-training. Clustering of one log profile is iteratively constrained by the similarities of the other profiles, leading to a unique clustering pattern.

To overcome the issue of heterogeneous scaled similarity criteria, occurring when multiple criteria are included, Delias et al. [6] propose an outranking approach. The authors build an overall metric using a non-compensatory methodology to overcome this issue. Song et al. [15] presented a comparative study to improve trace clustering using dimensionality reduction methods. The authors show the effect of applying three different reduction methods on the performance of trace clustering. To further improve the clustering result, De Koninck et al. [12] incorporate expert knowledge into the clustering to produce results that are more consistent with the expert's expectations. However, distance-based trace clustering do not consider the model evaluation bias, neglecting the accuracy of the reconstructed process models from the sub-logs.

Model-based trace clustering techniques such as [17, 21] combine the clustering bias and the model bias into an integrated view. ActiTraC [21] directly optimizes the fitness of the underlying process models to produce accurate results. In [17] a similar method is proposed which overcomes the stability issues of ActiTraC by first optimizing the average complexity of the models and then improve the accuracy of each model separately. While model-based approaches are good to produce accurate process models, they neglect the other process perspectives such as the data perspective.

The method presented in this paper aims at addressing both mentioned issues. It is inspired by our prior work [14] in which we cluster documents based on usage behavior and content into activity-centric groups. In this paper, we combine distance-based and model-based trace clustering methods to improve clustering results. We transfer the same idea to trace clustering in process mining, combining two distance measures to calculate the similarity between cases, including the control-flow and the data perspective. To retrieve accurate process models, we use an optimization algorithm to adjust the weighting of the distance measures and clustering parameters.

### 3 Hybrid Feature Set Clustering

In this section, we introduce our hybrid feature set clustering method. We extend existing trace clustering by optimizing the clusters using model fitness as a quality measure and incorporating case attributes.

#### 3.1 Notation

First, we introduce the notations that are used throughout this paper. They were derived from [1].

**Definition 1** (*Event, Attribute*). Let  $\mathcal{E}$  be the set of all possible event identifiers. Events may be described by attributes, such as the timestamp. Let  $\mathcal{A}$  be the set of attributes and  $\mathcal{V}_a$  the set of all possible values of attribute  $a \in \mathcal{A}$ . For an event  $e \in \mathcal{E}$  and an attribute  $a \in \mathcal{A}$ :  $\#_a(e)$  is the value of attribute  $a$  for event  $e$ .

**Definition 2** (*Case, Trace, Event Log*). Let  $\mathcal{C}$  be the set of all possible case identifiers. Cases can also have attributes. For a case  $c \in \mathcal{C}$  and an attribute  $a \in \mathcal{A}$ :  $\#_a(c)$  is the value of attribute  $a$  for case  $c$ . Each case contains a mandatory attribute trace:  $\#_{\text{trace}}(c) \in \mathcal{E}^*$ , also denoted as  $\hat{c} = \#_{\text{trace}}(c)$ .

A trace is a finite sequence of events  $\sigma \in \mathcal{E}^*$  such that each event only occurs once:  $1 \leq i < j \leq |\sigma| : \sigma(i) \neq \sigma(j)$ .

An event log is a set of cases  $L \subseteq \mathcal{C}$  such that each event only occurs at most once in the log.

**Definition 3** (*Classifier*). For an event  $e \in \mathcal{E}$ ,  $\underline{e} = \#_{\text{activity}}(e)$  the activity name of the event  $e$ . The classifier can also be applied to sequences  $\langle e_1, e_2, \dots, e_n \rangle = \langle \underline{e}_1, \underline{e}_2, \dots, \underline{e}_n \rangle$ .

Table 1 shows an example event log of a procurement process. The log consists of the cases  $L = \{1, 2\}$ , events  $E = \{11, 12, 13, 14, 21, 22\}$  and the attributes  $\mathcal{A} = \{\text{Case id, Event id, Vendor, Category, Timestamp, Activity, Resource}\}$ . The table also shows the values of the attributes, for example,  $\#_{\text{category}}(1) = \text{Office supplies}$  or  $\#_{\text{activity}}(13) = \text{PO created}$ .

**Table 1.** Simplified example event log of a procurement process.

Case id	Event id	Vendor	Category	Timestamp	Activity	Resource
1	11	B. Trug	Office supplies	2017-04-17 10:11	PR created	John
1	12			2017-04-18 14:55	PR released	Maria
1	13			2017-04-18 17:12	PO created	Roy
1	14			2017-04-29 09:06	Goods receipt	Ryan
2	21	Company	Computer	2017-04-19 17:45	PO created	Emily
2	22			...	...	...

### 3.2 Our Approach

Most trace clustering methods define a similarity function between the event sequence of cases (e.g., Levenshtein distance or bag-of-activities) and then apply a clustering algorithm (e.g., k-means, hierarchical, or partition methods) to segment the event log. As a result, such methods provide a set of sub-logs which contain maximized intra-cluster and minimized inter-cluster similarity, neglecting the quality of the underlying process model [4, 21]. This may lead to unsatisfactory results of the reconstructed models. Another issue of most existing trace clustering approaches is that they do not explore the relationships between case attributes to identify different behaviors on the data perspective. However, cases might be influenced by their data attributes. For process analysts, it might also be interesting to separate similar traces and put them into a different cluster if their corresponding data attributes are inconsistent.

Our approach addresses both issues. Instead of solely relying on the similarity function between traces, we additionally use the fitness [20] of the underlying process model as a criterion for quality of sub-logs. The fitness of a model is a normalized measure reflecting how many behaviors featured in the event log are also contained in the discovered process model. Our goal is to find an optimal separation of the event log such that the different process behaviors on both perspectives are clustered separately, while optimizing the fitness of the underlying process models. Additionally, the number of clusters should be kept reasonably small. Including additional data attributes uncovers certain behavior, for example, different levels of product quality checks, which would normally be clustered together despite being completely different regarding the data attributes. By adding this additional perspective, we are able to separate such behaviors, despite the cases being similar with respect to the control-flow. We combine both perspectives to extract more valuable knowledge about the execution of processes allowing us to distinguish between the different process behaviors more accurately.

In the following, we will describe our hybrid feature set clustering approach in detail.

---

**Algorithm 1.** Algorithm to retrieve the clusters
 

---

- 1 Let  $L$  be the event log, and let  $\hat{L} = \{\underline{(\hat{c})} : c \in L\}$  be the set of distinct event traces of  $L$ .
- 2 Define  $lev(x, y)$  to be the edit distance of the event traces  $x, y \in \hat{L}$ .
- 3 Define  $sim_{lev}^*(X, Y)$  to be the edit distance between two sets of event traces  $X, Y \subseteq \hat{L}$ :

$$sim_{lev}^*(X, Y) = \sum_{x \in X} \sum_{y \in Y} lev(x, y) / (|X| \cdot |Y|)$$

- 4 Define  $cases(t) = \{c : c \in L \wedge (\hat{c}) = t\}$  as the cases following event trace  $t \in \hat{L}$ .
- 5 Define  $encode(c) = \{\mathcal{I}_a(\#_a(c)) : a \in \mathcal{A}\}$  with  $c \in L$  and  $\mathcal{I}$  as an integer index function; further define  $encodes(C) = \{encode(c) : c \in C\}$ .
- 6 Let  $\mathcal{S}$  be the universe of all possible itemsets,  $S \subseteq \mathcal{S}$  and  $s_i$  being the  $i$ -th itemset in  $\mathcal{S}$ .
- 7 Define  $itemsets : \hat{L} \rightarrow \mathcal{P}(\mathcal{S})$  as the function that returns the frequent itemsets using the *FPclose* algorithm with  $\theta$  being the *minimum support threshold*:

$$itemsets(t) = FPclose(encodes(cases(t)), \theta)$$

- 8 Define  $sim_{itemsets}(S_a, S_b)$  to be the similarity function of the itemsets with  $S_a, S_b \subseteq \mathcal{S}$ :

$$sim_{itemsets}(S_a, S_b) = \frac{2 \cdot |S_a \cap S_b|}{|S_a| + |S_b|}$$

- 9 Define  $traces(s) = \{t : t \in \hat{L} \wedge s \in itemsets(t)\}$  with  $s \in \mathcal{S}$  to be the inverse function of *itemsets* which returns the traces for a given itemset.
- 10 Define  $sim(s_a, s_b)$  to be the combined similarity function with  $s_a, s_b \in \mathcal{S}$ ,  $w \in \mathbb{R}$  and  $0 \leq w \leq 1$  to be the weighting factor:

$$sim(s_a, s_b) = w \cdot sim_{lev}^*(traces(s_a), traces(s_b)) + (1 - w) \cdot sim_{itemsets}(s_a, s_b)$$

- 11 Define  $M : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$  to be the itemset distance matrix

$$M = (m_{ij}) = sim(s_i, s_j)$$

- 12 Let  $cluster(M, n)$  be the hierarchical clustering function that returns the cluster index of each itemset as a vector of size  $|\mathcal{S}|$  with  $n \in \mathbb{N}$  as the number of clusters to generate.
- 13 Define  $C(k)$  to be the set of the traces in cluster  $k$

$$C(k) = \{cases(traces(s_j)) \mid s_j \in \mathcal{S} \wedge c_j \in cluster(M) \wedge c_j = k\}$$


---

**Candidate Clusters.** The first step of our approach is to generate a candidate set of clusters. Algorithm 1 shows the generation of the clusters using the combined similarity measure. From the event log  $L$  we extract all distinct event traces  $\hat{L}$  (Line 1). For comparing the traces, we use the Levenshtein edit distance. It is

defined as the minimum number of edit operations that are required to transform one sequence into another. The edit operations are insertion, deletion, or substitution of an element in the sequence. Each operation has a cost of 1. We denote the normalized Levenshtein edit distance between two traces  $x, y \in \hat{L}$  as  $lev(x, y)$  (line 2). In line 3, we additionally define a function  $sim_{lev}^*$  which calculates the pairwise normalized Levenshtein distance between the traces in two sets. It is noteworthy that the Levenshtein distance can be replaced by a more advanced measure (e.g., one that also recognizes concurrency).

As a second similarity measure, we incorporate the attributes and their values of a case by extracting further knowledge about the underlying case relations. The idea is to use the case attributes in the event log to extract dependencies between attributes in certain process behaviors. Consider the procurement of supplies. Usually, there are different order approval steps involved depending, for example, on the material type of the purchased item. So, for office supplies there might be only one approval step whereas for the spare part of an expensive machine multiple approval steps of different departments are required. Such variations are usually deployed to reduce the amount of process steps. With the use of the case attributes, our approach is able to distinguish such behaviors even if the behavior on the control-flow perspective is very similar.

To extract such knowledge patterns from case attributes, we use frequent itemset mining (line 5–7). Specifically, we use the FPclose algorithm [8] to extract closed frequent itemsets to limit the number of itemsets. An itemset is closed if there exists no suitable superset which has the same support. We calculate the frequent itemsets for all cases that follow the same trace  $t \in \hat{L}$ . To retrieve all cases that follow a specific trace  $t$ , we define a function  $cases(t)$  (line 5) which maps a given event trace  $t$  to their respective cases based solely on the event sequence. Note that while  $cases(t)$  yields a set of cases with identical behavior, their case attributes might be quite different for which we mine frequent itemsets. We calculate the frequent itemsets for a given *minimum support threshold*  $\theta$ . So, attribute-value pairs that occur in a certain amount of cases are extracted as frequent itemsets, directly taking the frequency of cases following the same trace into account. Case attributes are transformed using integer encoding, which is a mapping  $\mathcal{I}_a : \mathcal{V}_a \rightarrow \mathbb{N}$  (line 6), assigning each attribute-value pair a unique positive integer.  $encode(c)$  is the encoding function that encodes all attributes of a case  $c \in L$ . A similarity function between the two itemsets  $S_1, S_2 \subseteq S$  is defined in line 8. It compares the two itemsets, in particular, the attribute-value pairs, and returns the proportion of items which are contained in both sets.

In our approach, we do not cluster the cases itself but the itemsets of all cases that follow the same trace. We define a similarity function  $sim(s_a, s_b)$  that, on the one hand, calculates the similarity between itemsets and, on the other hand, compares the traces that share the same itemsets (line 10). With the weighting factor  $w$  we can control the balance between itemset similarity and trace similarity. It is noteworthy that even if  $w = 1$  the itemset similarity is indirectly incorporated because traces that share the same itemsets are merged together. Proceeding further, we generate a distance matrix  $M$  (line 11) and

use the *Agglomerative Hierarchical Clustering* algorithm to build a vector that contains the cluster index for each itemset (line 12). In line 13 the result is generated.  $C(k)$  contains a set of traces that are clustered into cluster  $k$ .

**Generating Non-overlapping Clusters.** Due to the construction of the Algorithm 1, generated clusters are overlapping. This is because we create the clusters based on the itemsets and not based on the cases. For all cases that follow a specific trace, multiple frequent itemsets can be mined which are not necessarily clustered together, for example if the distance to other itemsets is lower. Whenever this occurs, a trace and their corresponding cases are part of multiple clusters. Even if it might also be interesting to analyze overlapping clusters, in this paper we aim for non-overlapping clusters to reconstruct process models using a discovery algorithm. To resolve the overlapping clusters, we assign traces that are assigned to multiple clusters to the one with the minimum distance with respect to sequence similarity.

We denote  $buildCluster(\theta, n, w)$  to be the function which executes Algorithm 1 to generate the candidate clusters and the algorithm to resolve the overlapping.

**Determine Optimal Parameters.** In the last step, we optimize the minimum support threshold  $\theta$ , the number of clusters  $N$ , the weighting factor  $w$  such that the fitness of the underlying model is maximized. The goal is to find an optimal separation of cases such that the different behaviors are separated into clusters while still being able to reconstruct accurate process models. We use the *Flexible Heuristics Miner* [22] to reconstruct the models for each cluster because of its low computational costs and high accuracy in real-life scenarios. From these models we calculate the weighted average improved continuous semantics fitness measure (ICS) over all models.

**Definition 4 (Weighted ICS Fitness).** Let  $ics_k$  the ICS-Fitness of a model  $k$  and  $n_k$  the number of cases in  $k$ , then the weighted ICS Fitness is defined as:

$$ICS - Fitness = \frac{\sum_{k=1}^N (n_k \cdot ics_k)}{|L|}$$

Besides the weighted fitness of the models, we also optimize the number of cases assigned to a cluster, the number of clusters and the cluster silhouette coefficient. It might occur that  $\theta$  is chosen too high such that a small amount of frequent patterns were extracted which should be avoided.

We use the *Particle Swarm Optimization* (PSO) [11] algorithm to maximize the fitness of the mined models of each cluster, finding optimal values for  $\theta$ ,  $n$ , and  $w$ . PSO is an evolutionary optimization algorithm which was initially inspired by bird flocking, specifically, the group dynamics of the bird behavior. PSO maintains a swarm of  $n$  particles  $\mathbf{p}$ , the candidate solutions, which move around in the search-space. Initially, particles  $\mathbf{p}_0$  are randomly distributed in the search



space and assigned an initial movement velocity  $\mathbf{v}_0$ . Each particle maintains, the inertia  $\omega$ , the best known position  $\mathbf{p}_{best}$ , the global best position over all particles  $\mathbf{g}_{best}$ , a cognitive weighting factor  $c_k$  and a social weighting factor  $c_s$ .

**Definition 5.** For each iteration, a new velocity vector  $\mathbf{v}_{n+1}$  is calculated with  $r_1, r_2$  being random factors for each iteration:

$$\mathbf{v}_{n+1} = \omega \cdot \mathbf{v}_n + c_k \cdot r_1 \cdot (\mathbf{p}_{best} - \mathbf{p}_n) + c_s \cdot r_2 \cdot (\mathbf{g}_{best} - \mathbf{p}_n)$$

The movement of the particles is determined by their current position and velocity as well as the local and global best known positions. Particles are moved until the maximum number of iterations is reached. PSO executes the *buildCluster*( $\theta, N, w$ ) function and optimizes the model fitness as well as the proportion of assigned traces. In our experiments, we found that 10 iterations with 5 particles are appropriate. Although PSO does not guarantee a global optimum, our evaluation results suggest that even local optima yield good results.

## 4 Evaluation

In this section, we evaluate our proposed approach in two different evaluation settings. First, synthetic event logs are used to evaluate the quality of the generated clusters based on well-known evaluation measures for cluster analysis as well as process mining related measures. We compare our approach with six other clustering methods of the related work. Secondly, we use real-life event logs to show the applicability of our approach. Here, we focus on the quality of the generated models with respect to comprehensibility and accuracy. Our proposed approach is implemented as the *HybridCluster plugin*<sup>1</sup> in ProM.

### 4.1 Synthetic Event Logs Evaluation

We use synthetic event logs to evaluate and compare the performance of our clustering approach with respect to the quality of the generated clusters.

**Datasets.** Currently, there exists no comprehensive benchmark for the evaluation of trace clustering in the related work that focuses on the different behaviors on both the control-flow and data perspective. We generated synthetic event logs from random process models of different complexity (varying number of activities, maximum depth and branching factor). Five process models (see Table 2) are generated using PLG2 [5]: Small, Medium, Large, Huge, Wide and custom designed model with human readable activity names, all derived from [13].

For a representative data perspective, we generate sets of possible attribute values  $\mathcal{V}_a$ , of size 20, for each of the case attributes  $a \in \mathcal{A}$ . Then we assign case attribute values to all cases by sampling from  $\mathcal{V}_a$ , for each attribute  $a$ . To introduce some causal relationships, we force certain combinations of attribute values

<sup>1</sup> Source code available at: <https://github.com/alexsee/HybridClusterer>.

**Table 2.** Process models used for generating the event log: Number of activity types (# at), number of transitions (# tr), number of variants (# dpi), maximal trace length and out-degree.

Model	# at	# tr	# dpi	max length	out-degree
P2P	14	16	6	9	1.14
Small	22	26	6	10	1.18
Medium	34	48	25	8	1.41
Large	44	56	28	12	1.27
Wide	56	75	39	11	1.34
Huge	36	53	19	7	1.47

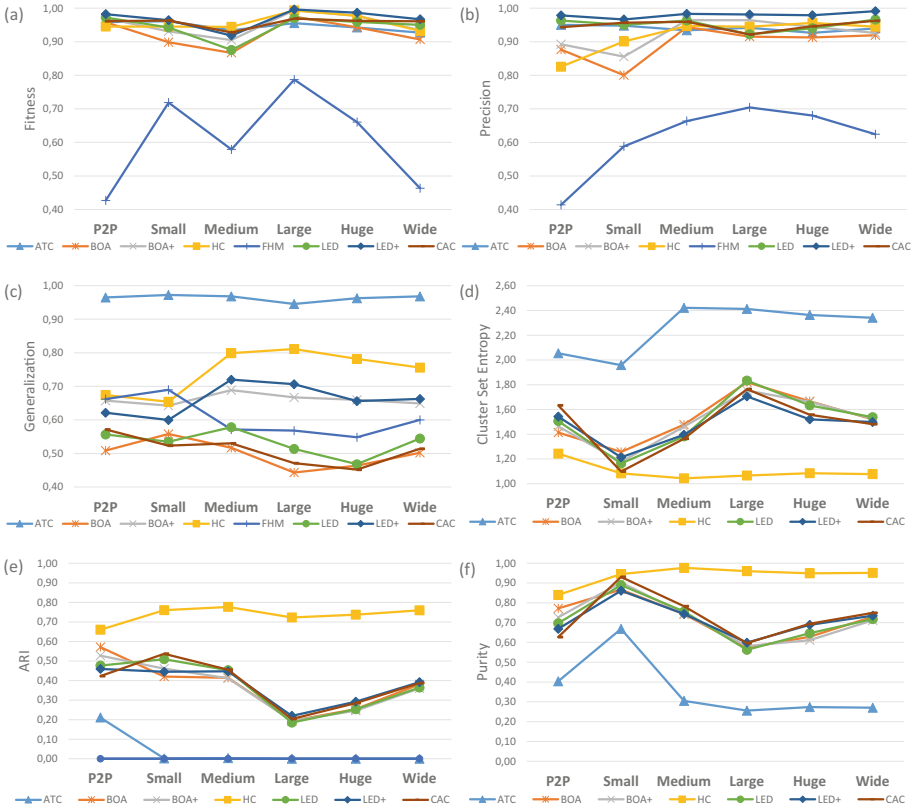
to occur more frequently than others depending on the event sequence of a case. Hence, each sequence of events will have certain attributes and attribute values that represent causalities for this sequence. Note that it is possible that multiple event sequences feature the same attribute value patterns. Consequently, we obtain patterns that correlate both with the data perspective and the control-flow, which will then represent our ground truth for the clusters. In the evaluation we generate five clusters consisting of the different generated patterns.

To increase the complexity of the task we also incorporate some level of noise (ranging from 0.0 to 0.2) into the control-flow [13]. For example by perturbing the order of events, skipping of events or executing multiple events. In summary, we generated event logs with different sizes (1000, 2000, 5000 and 10000), varying number of attributes (5, 10, 15, 20) and 3 different noise levels (0.0, 0.1, 0.2), resulting in 288 event logs<sup>2</sup>.

**Accuracy Results.** We compare our hybrid feature set clustering approach (HC) with six other trace clustering methods: bag-of-activities (BOA) [4], Levenshtein edit distance (LED) [4], Context-Aware-Clustering (CAC) [3] and Acti-TraC (ATC) [21]. For BOA and LED agglomerative hierarchical clustering with ward linkage is used. For BOA+ and LED+ we filter out the event sequences that occur less than 2 times. BOA, BOA+, LED, LED+ and CAC do not provide any optimization to find the optimal number of clusters, thus we vary the number of clusters from 2 to the number of distinct traces and show the best results for the same or less clusters as HC. For ATC we use standard setting, 80% stopping criterion for the frequency-based and MRA distance-based selective sampling. As a baseline without clustering (FHM) and for generating the process models of each cluster, we use the Flexible Heuristics Miner [22].

*Weighted Fitness, Precision and Generalization.* To evaluate the accuracy of the discovered models, we report the ICS fitness of the FHM which is in the range of  $(-\infty, 1]$ . For precision and generalization, we first use the *Heuristics*

<sup>2</sup> Models are openly available: <https://doi.org/10.7910/DVN/QBL1K0>.



**Fig. 1.** Evaluation results of the synthetic event logs: (a) weighted fitness of the models; (b) weighted precision of the process models; (c) weighted generalization of the models; (d) cluster set entropy (lower is better); (e) adjusted rand index; (f) purity.

*Net to Petri Net* plugin in ProM and then calculate the measures introduced in [19]. Note that precision and generalization are sensitive measures that strongly depend on the petri net (e.g., if the petri net contains silent or duplicate transitions). While the weighted fitness (see Fig. 1(a)) of the process model without any clustering is quite low for all evaluated event logs, all trace clustering method produced good results within the range of 0.969 and 0.925. The best ICS fitness was achieved by LED+, followed by CAC and our HC method. Comparing the precision (see Fig. 1(b)) of the models reveals that LED+ slightly outperformed all other methods, followed by LED, ATC and BOA. Our HC method achieved an average precision of 0.920 compared to 0.980 of the best. However, it is still significantly better than the FHM. With respect to generalization (see Fig. 1(c)) ATC outperformed all methods significantly, whereas our HC method is the second best.

**Table 3.** Performance of the related work and our hybrid cluster approach with respect to process model and clustering evaluation; best values in bold typeface.

	Process model			Clustering			
	Fitness	Precision	Generalization	Purity	ARI	CSE	C
FHM [22]	0.606	0.612	0.607	-	-	-	1.0
ActiTraC [21]	0.946	0.940	<b>0.964</b>	0.363	0.036	2.258	1.8
Bag-of-activities [4]	0.925	0.895	0.499	0.719	0.372	1.526	17.5
Bag-of-activities+	0.952	0.925	0.661	0.713	0.367	1.507	20.1
Levensthein [4]	0.945	0.951	0.533	0.711	0.374	1.509	21.9
Levensthein+	<b>0.969</b>	<b>0.980</b>	0.661	0.716	0.376	1.480	18.6
CAC [3]	0.957	0.948	0.510	0.729	0.381	1.484	22.6
Hybrid clusterer	0.956	0.920	0.746	<b>0.937</b>	<b>0.736</b>	<b>1.100</b>	26.7

*Cluster Set Entropy, Adjusted Rand Index and Purity.* While fitness, precision and generalization evaluate the accuracy of the process models, cluster set entropy (CSE), adjusted rand index (ARI) and purity (see Fig. 1 (d)–(f) and Table 3) evaluate the calculated clustering against the ground truth. Our approach outperforms all other methods over all event logs with an average CSE of 1.100, an average ARI of 0.736 and an average purity of 0.937. A clear ranking of the other methods cannot be made, as the field is mixed here. For smaller models, the performance difference between our HC and the related work is less significant.

*Discussion.* In our experiments, the HC approach is the only method that incorporates case attributes. Hence, the existence of case attributes that are somehow related to the traces is essential for our method. While we used synthetic event logs that contain such relationships, other event logs may not have these relationships. Here, other trace clustering methods may perform better. Still, the aim of our method is to combine both perspectives, the control-flow and the case attributes, to generate behavioral similar sub-logs. Our method tries to overcome the issue of unrelated case attributes by optimizing the balance between both perspectives. In cases where no relationship is found, our method will prefer giving the control-flow similarity more contribution.

However, when such a relationship exists, as presented in our evaluation, our method provides a solid separation of the event log. With respect to CSE, ARI and purity, our method works better than control-flow only based methods. Our method is also able to generate process models with a good fitness, precision and generalization, although being outperformed by other methods. The variance of the traces within a cluster might be high which negatively influences the fitness, precision and generalization. Another observation is that our approach generates more clusters than other methods (see Table 1). This is mainly caused by the separation of the different behaviors on the different perspectives.

**Table 4.** Overview of real-life logs: The number of process instances (# pi), number of events (# ev), number of activity types (# at) and the number of variants (# dpi).

Event log	Event log properties				Description
	# pi	# ev	# at	# dpi	
P2P	33 277	255 427	37	7 026	Procurement process
EV	1 434	8 577	27	116	Case handling system
HOSBILL	100 000	451 359	18	1 020	Hospital invoice billing
HOSLOG	1 143	150 291	624	981	Case handling in hospital
ROAD	150 370	561 470	11	231	Road traffic fine process

## 4.2 Real-Life Event Logs Evaluation

The second part of the evaluation applies our approach to real-life event logs. In Table 4 we show some basic statistics of the used event logs, originated from different environments to show the applicability of our approach in various scenarios. All event logs except for P2P are openly available<sup>3</sup>. Because we use real-life event logs for which we do not know their real behavior, we focus on the evaluation of the following measures: First, we measure the weighted fitness, precision and generalization of models discovered by the FHM of each cluster. Again, the heuristics nets are converted using the ProM plugin as mentioned before. Second, we measure the complexity of the resulting models. For the five real-life experiments we use the same settings as used in the synthetic evaluation. Again, for BOA, BOA+, LED and LED+ we only report the best fitness for the same or less number of clusters as HC.

**Accuracy Results.** The weighted fitness, precision and generalization results are presented in Table 5. In all cases the FHM was outperformed by all clustering techniques, concluding that a single model is not sufficient to accurately model the observed behavior. The vector-based clustering approaches, i.e., BOA, BOA+, LED and LED+, provide a relatively good fitness of the models. Acti-TraC in general provides better fitness values as the vector-based approaches. HC provides better or similar results as other clustering methods, except for HOSLOG. This is due to the fact that HOSLOG contains many unique traces. The HOSLOG event log originates from an hospital where the examination of each patient is recorded and contains cases that usually do not follow a strictly defined sequence. With respect to the precision and generalization, the field is mixed.

<sup>3</sup> [http://data.4tu.nl/repository/collection:event\\_logs\\_real](http://data.4tu.nl/repository/collection:event_logs_real).

**Table 5.** Results of the real-life logs showing average weighed fitness. Precision and generalization in parenthesis. Missing values due to canceled calculation after 12 h.

	P2P	EV	HOSBILL	HOSLOG	ROAD
FHM [22]	-0.799	0.649	-0.781	0.554	0.434
	(0.15/0.27)	(0.66/0.41)	(0.63/0.50)	-	(0.98/0.39)
ActiTraC [21]	<b>0.729</b>	0.893	0.651	<b>0.719</b>	0.973
	(0.70/0.81)	(0.72/0.91)	-	-	(0.99/0.93)
Bag-of-activities [4]	0.146	0.793	-0.354	0.346	0.755
	(0.29/0.48)	(0.70/0.37)	(0.86/0.37)	-	(0.95/0.50)
Bag-of-activities+	0.519	0.839	-0.134	0.685	0.813
	(0.50/0.77)	(0.99/0.36)	(0.98/0.73)	(0.42/0.88)	(0.90/0.65)
Levensthein [4]	0.196	0.871	-0.291	0.406	0.837
	(0.42/0.55)	(0.70/0.47)	(0.77/0.62)	-	(0.99/0.72)
Levensthein+	0.725	0.857	-0.017	0.001	0.979
	(0.68/0.79)	(0.84/0.73)	(0.93/0.64)	(0.68/0.68)	(0.99/0.98)
CAC [3]	0.121	0.739	0.839	0.198	0.887
	(0.40/0.53)	(0.77/0.39)	(0.74/0.72)	-	(1.00/0.56)
Hybrid clusterer	0.723	<b>0.975</b>	<b>0.958</b>	0.515	<b>0.993</b>
	(0.61/0.77)	(0.96/0.55)	(0.97/0.81)	(0.88/0.86)	(0.99/0.99)

**Complexity Results.** We calculate four complexity measures based on related work [6, 18]. The graph density  $GD$  is defined as  $GD = \frac{|E|}{|N| \cdot (|N| - 1)}$  where  $|N|$  are the number of nodes in the discovered heuristics net and  $|E|$  the number of edges. The cyclomatic number  $CN$  is defined as  $CN = |E| - |N| + 1$ . The coefficient of connectivity  $CNC$  is defined as  $CNC = |E|/|N|$  and the coefficient of network complexity  $CNCK$  is defined as  $CNCK = |E|^2/|N|$ .

Table 6 shows the average and maximum of each complexity measure. We can see that our approach creates more clusters than other methods. This might be due to the fact that even similar traces are split into multiple clusters when they differ from the data perspective. The high number of clusters for ActiTraC is caused by the explosion in clusters for the P2P and the HOSLOG event log. When comparing the graph density, our method produces slightly denser process models than other methods. For cyclomatic number, coefficient of connectivity and coefficient of network complexity our hybrid clustering approach performs better than all compared methods. Comparing the average and the maximum numbers concludes that generated clusters of our method do not vary heavily.

**Table 6.** Complexity measures for each approach.

	$ C $	GD		CN		CNC		CNCK	
		avg	max	avg	max	avg	max	avg	max
ActiTraC [21]	198.4	0.15	0.66	11.10	116.60	1.38	2.22	46.37	477.28
Bag-of-activities [4]	18.2	0.12	0.20	82.33	114.20	1.15	1.20	329.05	456.40
Bag-of-activities+	17.6	0.24	0.51	2.84	5.40	0.78	1.00	12.06	20.00
FHM [22]	1	<b>0.10</b>	0.18	185.00	803.00	1.80	3.00	754.40	3258.00
Levensthein [4]	29.2	0.11	0.32	37.96	120.80	1.00	1.40	154.28	484.60
Levensthein+	22	0.20	0.42	3.61	7.60	0.76	1.20	15.10	28.80
CAC [3]	16.6	<b>0.10</b>	0.18	41.22	112.80	1.06	1.40	166.76	451.60
Hybrid clusterer	31.2	0.23	0.63	<b>2.15</b>	7.80	<b>0.67</b>	1.20	<b>10.62</b>	31.60

## 5 Conclusion

In this paper, we proposed a novel hybrid-feature set clustering approach in the area of process mining. While other trace clustering methods mainly rely on the control-flow, we use frequent pattern mining to extract further knowledge from the case attributes. To produce accurate process models, our method uses the particle swarm optimization to optimize the fitness of the underlying process models by automatically finding appropriate parameters.

We implemented our approach as an openly available ProM plugin. We evaluated our approach by conducting a comprehensive evaluation using synthetic and real-life event logs. We compared our approach to six other methods and showed that our approach is able to separate process behaviors on the control-flow as well as on the data perspective. For the synthetic event logs, our method reaches an ARI of 0.736 in average over all models evaluated, whereas the second best approach CAC reaches an ARI of 0.381. Besides using synthetic event logs, we used 5 real-life event logs to show the applicability of our method.

Something that we did not inspect is the question, if the identified clusters are relevant for process analysts during their analysis. Because this question is hard to answer without an extended user study, we would like to address this in future work. A first interview with a process mining consultant showed the interest in the idea of incorporating both perspective into the clustering. Additionally, it might also be interesting to incorporate expert knowledge to adjust the cluster quality. We also want to extend the approach to support numeric attributes and optimize runtime performance, because currently the algorithm has to perform certain operations multiple times which can be precalculated or cached.

Overall, we can conclude that our hybrid feature set clustering approach is a promising method for segmenting the event log into behavioral similar clusters.

**Acknowledgements.** This project [522/17-04] is funded in the framework of Hessen Modellprojekte, financed with funds of LOEWE, Förderline 3: KMU-Verbundvorhaben (State Offensive for the Development of Scientific and Economic Excellence), and by the German Federal Ministry of Education and Research (BMBF) Software Campus project “AI-PM” [01IS17050].

## References

1. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes, 2nd edn. Springer, Heidelberg (2011). <https://doi.org/10.1007/978-3-642-19345-3>
2. Appice, A., Malerba, D.: A co-training strategy for multiple view clustering in process mining. *IEEE Trans. Serv. Comput.* **9**(6), 832–845 (2016). <https://doi.org/10.1109/tsc.2015.2430327>
3. Bose, R.P.J.C., van der Aalst, W.M.P.: Context aware trace clustering: towards improving process mining results. In: Proceedings of the 2009 SIAM International Conference on Data Mining, pp. 401–412. Society for Industrial and Applied Mathematics (2009). <https://doi.org/10.1137/1.9781611972795.35>
4. Bose, R.P.J.C., van der Aalst, W.M.P.: Trace clustering based on conserved patterns: towards achieving better process models. In: Rinderle-Ma, S., Sadiq, S., Leymann, F. (eds.) BPM 2009 Workshops. LNBIP, vol. 43, pp. 170–181. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-12186-9\\_16](https://doi.org/10.1007/978-3-642-12186-9_16)
5. Burattin, A.: PLG2: multiperspective process randomization with online and offline simulations. In: CEUR Workshop Proceedings, vol. 1789, pp. 1–6 (2016)
6. Delias, P., Doumpos, M., Grigoroudis, E., Matsatsinis, N.: A non-compensatory approach for trace clustering. *Int. Trans. Oper. Res.* (2017). <https://doi.org/10.1111/itor.12395>
7. Evermann, J., Thaler, T., Fettke, P.: Clustering traces using sequence alignment. In: Reichert, M., Reijers, H.A. (eds.) BPM 2015 Workshops. LNBIP, vol. 256, pp. 179–190. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-42887-1\\_15](https://doi.org/10.1007/978-3-319-42887-1_15)
8. Grahne, G., Zhu, J.: Fast algorithms for frequent itemset mining using FP-trees. *IEEE Trans. Knowl. Data Eng.* **17**(10), 1347–1362 (2005). <https://doi.org/10.1109/tkde.2005.166>
9. Greco, G., Guzzo, A., Pontieri, L., Sacca, D.: Discovering expressive process models by clustering log traces. *IEEE Trans. Knowl. Data Eng.* **18**(8), 1010–1027 (2006). <https://doi.org/10.1109/tkde.2006.123>
10. Han, J., Cheng, H., Xin, D., Yan, X.: Frequent pattern mining: current status and future directions. *Data Min. Knowl. Disc.* **15**(1), 55–86 (2007). <https://doi.org/10.1007/s10618-006-0059-1>
11. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN 1995 - International Conference on Neural Networks. IEEE. <https://doi.org/10.1109/icnn.1995.488968>
12. De Koninck, P., Nelissen, K., Baesens, B., vanden Broucke, S., Snoeck, M., De Weerd, J.: An approach for incorporating expert knowledge in trace clustering. In: Dubois, E., Pohl, K. (eds.) CAiSE 2017. LNCS, vol. 10253, pp. 561–576. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-59536-8\\_35](https://doi.org/10.1007/978-3-319-59536-8_35)
13. Nolle, T., Seeliger, A., Mühlhäuser, M.: Unsupervised anomaly detection in noisy business process event logs using denoising autoencoders. In: Calders, T., Ceci, M., Malerba, D. (eds.) DS 2016. LNCS (LNAI), vol. 9956, pp. 442–456. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-46307-0\\_28](https://doi.org/10.1007/978-3-319-46307-0_28)



14. Seeliger, A., Schmidt, B., Schweizer, I., Mühlhäuser, M.: What belongs together comes together. Activity-centric document clustering for information work. In: Proceedings of the 21st International Conference on Intelligent User Interfaces - IUI 2016. ACM Press (2016). <https://doi.org/10.1145/2856767.2856777>
15. Song, M., Yang, H., Siadat, S.H., Pechenizkiy, M.: A comparative study of dimensionality reduction techniques to enhance trace clustering performances. *Expert Syst. Appl.* **40**(9), 3722–3737 (2013). <https://doi.org/10.1016/j.eswa.2012.12.078>
16. Song, M., Günther, C.W., van der Aalst, W.M.P.: Trace clustering in process mining. In: Ardagna, D., Mecella, M., Yang, J. (eds.) *BPM 2008 Workshops*. LNBP, vol. 17, pp. 109–120. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-00328-8\\_11](https://doi.org/10.1007/978-3-642-00328-8_11)
17. Sun, Y., Bauer, B., Weidlich, M.: Compound trace clustering to generate accurate and simple sub-process models. In: Maximilien, M., Vallecillo, A., Wang, J., Oriol, M. (eds.) *ICSOC 2017*. LNCS, vol. 10601, pp. 175–190. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-69035-3\\_12](https://doi.org/10.1007/978-3-319-69035-3_12)
18. Thaler, T., Ternis, S., Fettke, P., Loos, P.: A comparative analysis of process instance cluster techniques. In: Proceedings der 12. Internationalen Tagung Wirtschaftsinformatik, WI 2015, August, pp. 423–437 (2015)
19. Vanden Broucke, S.K., De Weerd, J., Vanthienen, J., Baesens, B.: Determining process model precision and generalization with weighted artificial negative events. *IEEE Trans. Knowl. Data Eng.* **26**(8), 1877–1889 (2014). <https://doi.org/10.1109/TKDE.2013.130>
20. Weerd, J.D., Backer, M.D., Vanthienen, J., Baesens, B.: A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs. *Inf. Syst.* **37**(7), 654–676 (2012). <https://doi.org/10.1016/j.is.2012.02.004>
21. Weerd, J.D., vanden Broucke, S., Vanthienen, J., Baesens, B.: Active trace clustering for improved process discovery. *IEEE Trans. Knowl. Data Eng.* **25**(12), 2708–2720 (2013). <https://doi.org/10.1109/tkde.2013.64>
22. Weijters, A.J.M.M., Ribeiro, J.T.S.: Flexible heuristics miner (FHM). In: 2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM). IEEE (2011). <https://doi.org/10.1109/cidm.2011.5949453>