Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Mining High-Quality Clustering Models for Explaining and Analyzing Quantitative Process Deviances

Francesco Folino, Massimo
Guarascio, Luigi Pontieri

RT-ICAR-CS-16-02 **Aprile 2016**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR)
– Sede di Cosenza, Via P. Bucci 41C, 87036 Rende, Italy, URL: *www.icar.cnr.it*
– Sezione di Napoli, Via P. Castellino 111, 80131 Napoli, URL: *www.icar.cnr.it*
– Sezione di Palermo, Viale delle Scienze, 90128 Palermo, URL: *www.icar.cnr.it*

Mining High-Quality Clustering Models for Explaining and Analyzing Quantitative Process Deviances

Francesco Folino, Massimo Guarascio, and Luigi Pontieri

ICAR Institute, National Research Council, Rende (CS), Italy
{ffolino,guarascio,pontieri}@icar.cnr.it

Abstract. Increasing attention has been paid recently to the problem of analyzing and explaining "deviant" process cases (stored in some given execution log), i.e. instances of the process that diverged from prescribed/expected/legal behavior (e.g., frauds, faults, SLA violations). Specifically, in many real settings, deviant cases are labelled with a numerical deviance measure, and the analyst is interested in obtaining a fine grain unsupervised classification of such deviances, which can help her recognize and explain different deviance scenarios. Unfortunately, current approaches to the explanation of process deviances rely on preliminary labelling the given cases as either "deviant" or "non-deviant", and then inducing a rule-based classifier for discriminating among these two classes. By converse, we here propose a novel method for discovering accurate and easily-interpretable deviance-aware clustering models, expressed in terms of readable clustering rules (i.e., propositional patterns over diverse case and event properties). In particular, we devised an algorithm that heuristically solves the problem of finding an optimal deviance-aware clustering model, effectively and efficiently, and equips each discovered cluster with summary information, which can be exploited to interactively provide the analyst with process maps and distribution charts effectively showing distinctive characteristics of the cluster. Tests on a real-world process logs confirmed the capability of the approach to find high-quality clustering models.

Keywords: Clustering, Deviance Explanation, Business Process Intelligence.

1 Introduction

The analysis and explanation of deviant cases (a.k.a. instances), stored in a a given execution log of a business process, has been getting growing interest, as a way to identify, and eventually solve, critical issues (like fraud, security breach, SLA violations etc.) that may affect the compliance/performance of the process. In many real application scenarios, such cases are labelled with a numerical deviance measure, e.g., quantifying their associated degree of severity/risk. Several approaches have been proposed in the literature to tackle the explanation of deviant instances. The prevalent strategy relies on setting a threshold for separating "normal" outcomes from "deviant" ones, so that a set of (human-readable) classification rules can be learnt (by reusing some existing machine learning method, e.g. [3]) for discriminating the two resulting classes of cases. As in many real settings there not one predefined way to set the deviance threshold and there is no guarantee that the discovered rules identify really meaningful and interesting deviance groups, it may happen that many iterations of such a label-and-induce analysis process are needed, before eventually obtaining useful deviance models.

As a matter of fact, we do believe that analysts are often more interested in directly obtaining a finer grain data-driven classification of deviant cases, which help them recognize and

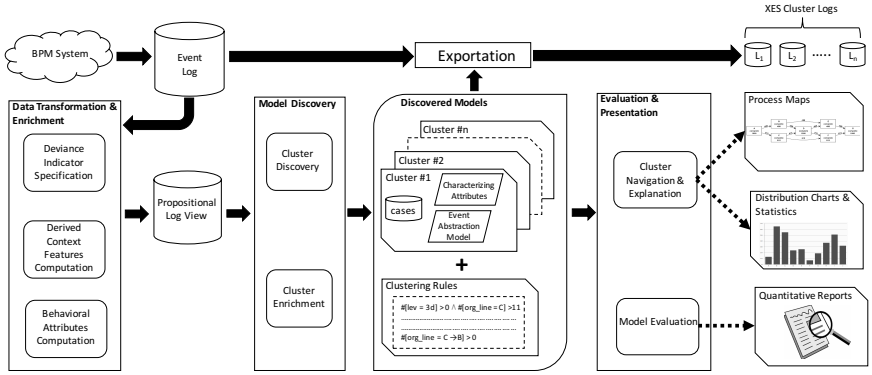


Fig. 1. Conceptual view of the proposed approach.

explain different deviance scenarios without any manual labelling step. Consequently, we want to address a more challenging task: automatically discover interesting data-driven deviance scenarios, while providing each with a readable explanation. In principle, one could think of addressing such a problem by leveraging off-the-shelf conceptual clustering techniques for inducing groups of deviant instances, such as those developed in the general field of *predictive clustering* [6, 2], while considering case deviance measures as target features. However, such methods (as shown in our experimentation) are likely to miss relevant deviant groups, due to their aim of minimizing intra-cluster variance (over the target features) for all of the discovered clusters (including those actually gathering non-deviant or lowly-deviant cases), as well as to their sensitiveness to (high-deviance) outliers.

To overcome the above limitations of current solutions, we here propose a novel approach to discovering accurate and easily-interpretable deviance-aware clustering models, expressed in terms of readable clustering rules (namely, propositional patterns over the non-target attributes of the given log cases), each of which is meant to capture and explain a distinguished deviance scenario. To this end, after formally stating the discovery of such a model as a novel (hard) optimization problem, we propose an algorithm that heuristically solves the problem in an effective and reasonably efficient way.

Besides extracting the very list of clustering rules (and the associated clusters), the proposed approach eventually provides two kinds of summary information for each of the discovered deviance clusters, say $\hat{c}$: *(i)* a graph-like process map (built upon an ad-hoc event abstraction function, specifically discovered for the cluster) that compactly describes the kinds of events happened when executing the cases in $\hat{c}$, and the main (possibly concurrent) flows of execution followed by these cases; and *(ii)* a ranked list of characterizing attributes for $\hat{c}$ (which may well go beyond the ones appearing in the clustering rule of $\hat{c}$), which can be used to produce comparative charts for the distribution of case attributes.

The proposed framework hence makes an original end-to-end framework (sketched in Fig. 1) for analyzing quantitative business-process deviances, which allows the analyst to easily explore each deviance cluster, and interactively visualize highly informative process maps and charts (like those shown in Figures 3 and 4, resp.), which can effectively help the analyst comprehend the behavior and properties of the discovered deviance scenarios.

We finally observe that one could think of addressing the above problem by adopting a *subgroup discovery* approach, where the goal is to discovery a number of descriptive patterns identifying each an instance group with a distribution of the given target concept/measure that substantially differs from the global one [1, 8]. However, as discussed

briefly in the concluding section, these approaches generate overlapping subgroups of data that avoid the precise identification (and explanation) of a unique deviant scenario.

2 Log Data and Deviances

The starting point of our analysis is a log L storing information on past executions of a business process. Conceptually, we can regard L as a set of *cases*, each of which corresponds to one process instance, and stores several data properties over a set $\mathcal{A}$ of (case) attributes —each attribute $A \in \mathcal{A}$ is a function $A : L \rightarrow Dom(A)$ that maps any case c to a value in the domain of A [1].

Besides such "flat" data, a log also stores a set of *traces*, providing each detailed information on the steps performed along the execution of the respective process instance. Let $trace(c) \in \mathcal{E}^*$ be the trace associated with any c in L, where $\mathcal{E}$ is the universe of events that the process may generate. For any such a trace τ, let $len(\tau)$ be the number of events in τ, and $\tau[i]$ be the i-th event of τ, for $i \in \{1, \ldots, len(\tau)\}$. Since, usually, log events gather multiple kinds of information (concerning, e.g., the executed activity, who executed it, etc.), and can be encoded as tuples over a set $\mathcal{A}_{\mathcal{E}}$ of event attributes, which, w.l.o.g., we assume to be categorical (possibly after pre-processing).

Finally, let $traces(L)$ be the set all the traces associated with the L's cases, and $events(L) \subseteq \mathcal{E}$ be the set of events that appear in some of these traces. For any $c \in L$ (resp., $e \in events(L)$, let $prop(c)$ (resp., $prop(e)$) be a tuple encoding its data properties in terms of attributes in $\mathcal{A}$ (resp., $\mathcal{A}_{\mathcal{E}}$).

Deviance indicators We assume that some deviance indicator is defined for the log cases, in the form of a function $\delta : L \rightarrow [0,1]$. Such a function maps any case c in L to a deviance score $\delta(c)$ in the range $[0,1]$ of real numbers, which quantifies how much c deviates from some reference model representing the normal/expected/prescribed behavior of the process. In many real-world settings, numeric deviance score are defined upon some case-oriented cost measure, say ω, such as the (total or utilization) time spent to handle a process instance, or the estimated risk that it infringed privacy/security policies. In such a case, a basic deviance indicator for ω can be defined w.r.t. some given reference value $\overline{r}$ (possibly defined in a data-driven way, as the minimum, mean, maximum of the scores) as : $\delta(c) = [\max(\omega(c) - \overline{r}, 0)]/[\max_{c' \in L} |\omega(c') - \overline{r}|]$.

Deviant vs. normal cases As mentioned in Sect. 1, current approaches to the explanation of process deviances rely on extracting a collection of classification (human-readable) rules that discriminate deviant cases from the other ones, once labelled any case c as deviant iff its $\delta(c)$ overcomes a threshold γ indicated by the analyst. We name such a case a γ-*deviant* case (or γ-*deviance*), and denote the set of all the γ-deviant cases in L by $L^{>\gamma}$. The underlying problem faced in the literature then amounts to inducing a binary (supervised) classification model, possibly using any standard machine learning methods, using all information available for a case as input data features.

3 Deviance-Aware Conceptual Clustering: Formal Framework

In order to address the peculiar clustering problem introduced in Section 1, we devise an ad-hoc conceptual clustering model, where each of the discovered clusters (but the last, gathering non-deviant cases) is meant to capture a relevant subset of γ-deviances, and is associated

[1] For the sake of brevity, we assume that $Dom(A)$ actually is the active domain of A w.r.t. L.

with a (conjunctive) propositional pattern, acting indeed as a clustering rule —notice that hereinafter the terms "pattern" and (clustering) "rule" will be used interchangeably. More specifically, final aim of our approach is to discover a conceptual clustering model that mainly consists of a list of such clustering rules, by regarding each case c in the given log L as a training tuple, represented over a set $\mathcal{A}^+$ of case attributes, which extends the set $\mathcal{A}$ originally available in L with a additional trace-based and context-based properties, computed automatically as discussed in the next section. All the "descriptive" attributes of the cases can be used to eventually define the clustering rules, while using each deviance scores $\delta(c)$ as a target attribute biasing the learning process towards "interesting" clustering models.

Clustering rules A *pattern* ϕ over the (extended) set $\mathcal{A}^+$ of case attributes can be seen as a set $\phi = \{\xi_1, \ldots, \xi_m\}$, where, for $i \in \{1, \ldots, m\}$, ξ_i is a constraint of the form $\xi_i : Dom(A) \to \{\texttt{true}, \texttt{false}\}$, restricting the range of values that A may take. Details on the constraints used in our approach are given in Sec. 4. The *cover* of ϕ over L, denoted by $cov(\phi, L)$ is the (multi-)set of L's tuples that satisfy all of the constraints in ϕ.

A *conceptual clustering model* for L (w.r.t. $\mathcal{A}^+$) is a list $\mathcal{M} = [\phi_1, \ldots, \phi_k]$ of patterns (over $\mathcal{A}^+$), for some $k \in \mathbb{N}^+$, encoding each a distinguished clustering rule (similarly to the decision lists of certain rule-based classifiers). For any case $c \in L$ and any $i \in \{1, \ldots, k\}$, it is $clusterID_{\mathcal{M}}(c) = i$ iff $c \in cov(\phi_i, L)$ and $c \notin cov(\phi_j, L)$ for any other index j such that $0 < j < i$. Moreover, $cluster_{\mathcal{M}}(L, i)$ is the i-th cluster produced by $\mathcal{M}$, i.e. $cluster_{\mathcal{M}}(L, i) = \{c \in L \mid clusterID_{\mathcal{M}}(c) = i\}$. To have a complete partitioning of L, we hereinafter assume that the last pattern in the list does not state any real constraint (i.e. $\phi_k = \emptyset$), hence acting as an "immaterial" default rule gathering non-deviant cases.

Deviance-Aware Clustering Models In our deviance-centric setting, a clustering model is expected to identify interesting groups of deviant cases, provided that there exist a list of propositional patterns that can separate them. To quantify the level of interestingness of a cluster (as representative of some relevant deviance scenario), we extend the notion of deviance indicator to clusters, by defining a function $\mathscr{I} : 2^L \to [0, 1]$ that maps any cluster $\hat{c}$ to a (average) deviance score, simply computed as $\mathscr{I}(\hat{c}) = |\hat{c}|^{-1} \cdot \sum_{c \in \hat{c}} \delta(c)$.

We also assume that the analyst may want to state three kinds of requirements for every deviance cluster: *(i)* a lower bound γ for the deviance score, *(ii)* a minimal cluster size σ, and *(iii)* a maximal length *maxLen* for the associated rule/pattern. The kind of clustering models constituting the search space of our learning task is defined next.

Definition 1 (DACM). Let $\gamma \in [0, 1)$, $\sigma \in \mathbb{N}^+$ and $maxLen \in \mathbb{N}^+ \cup \{\infty\}$ be three thresholds stating the three kinds of requirements cited above. Then, a *Deviance-Aware Clustering Model* (short DACM) $\mathcal{M}$ (for log L and deviance indicator δ) w.r.t. γ, σ and *maxLen* [2] is a list $\mathcal{M} = [\langle \phi_1, \hat{c}_1, \alpha_1, AList_1 \rangle, \ldots, \langle \phi_k, \hat{c}_k, \alpha_k, AList_k \rangle]$ such that:

(i) $[\phi_1, \ldots, \phi_k]$ is a conceptual clustering model for L (with $\phi_k = \emptyset$);
(ii) for each $j \in \{1, \ldots, k\}$, it is $\hat{c}_j = cluster_{\mathcal{M}}(L, j)$;
(iii) for each $j \in \{1, \ldots, k-1\}$ it holds $\mathscr{I}(\hat{c}_j) > \gamma$, $|(\hat{c}_j)| \geq \sigma$ and $|\phi| \leq maxLen$;
(iv) for each $j \in \{1, \ldots, k-1\}$, α_j is an event abstraction function that maps each event in $\mathcal{E}$ to a label representing a distinguished class of event/activity, while $AList_j$, is an ordered list of case attributes that are distributed differently in $\hat{c}_j$ than in the rest of the log (the higher the difference, the earlier the attribute features in the list). □

Notably, the description provided by such a model for the discovered deviance clusters goes beyond the mere list of clustering rules, which may sometimes be a little difficult to interpret, and give a partial explanation for deviant behaviors —indeed, each rule just describes

[2] For better readability, we will omit $L, \delta, \gamma, \sigma$ and *maxLen* when they are clear from the context.

a (locally) minimal subset of characteristics allowing to discriminate a cluster from the remaining ones, disregarding other distinguishing features of the cluster. The abstraction function α_j and attribute list $AList_j$ are right meant to fill this gap, by allowing to eventually build an effective visualization of the clusters' behavior, in terms of graph-like process maps and of comparative charts for the distribution of case attributes. In general, event abstraction is fundamental to obtain a meaningful process map, synthetically describing which kinds of events were generated by the process, and which were the main (possibly concurrent) flows of execution (across these kinds of event) that the analyzed process instances followed. In particular, each abstraction function α_j above is specifically meant to emphasize the behavioral differences (in terms of event flows) between the cases of $\hat{c}_j$ and the remaining ones. As discussed in Sect. 4, we concretely build α_j upon a conceptual (event) clustering model $\{\phi_1^E, \ldots, \phi_q^E\}$, such that $\phi_1^E, \ldots, \phi_q^E$ are mutually-exclusive patterns over $\mathcal{A}_E$.

4 Problem Statement and Solution Approach

Before precisely stating the problem faced in this work, let us introduce an ad-hoc function measuring the interestingness of any DACM $\mathcal{M}$, discovered for a log L with minimum deviance threshold $\gamma \in [0,1)$:

$$Q(\mathcal{M}, L^{>\gamma}) = \sum_{c \in L^{>\gamma}} w(c, L^{>\gamma}) \cdot \mathscr{I}(cluster_{\mathcal{M}}(c)) \tag{1}$$

where $w(c, L^{>\gamma})$ is a weight assigned to any γ-deviant case c, based on its respective deviance rank (the lower the rank, the higher the weight), and $\mathscr{I}(\hat{c})$ is the deviance score of cluster $\hat{c}$ (i.e. the average deviance score of $\hat{c}$'s cases). For the sake of concreteness, we hereinafter assume that $w(c, L^{>\gamma}) = |L^{>\gamma}| + 1 - rank(\delta(c))$, for any c in L.

The discovery of an optimal DACM is then stated as follows: **Given** A log L with an associated deviance indicator $\delta : L \rightarrow [0,1]$, and thresholds $\gamma \in [0,1)$, $\sigma \in \mathbb{N}^+$ and $maxLen \in \mathbb{N}^+ \cup \{\infty\}$; **Find** A DACM $\mathcal{M} = [\langle \phi_1, \hat{c}_1, \alpha_1, caseAttList_1 \rangle, \ldots, \langle \phi_k, \hat{c}_k, \alpha_1, caseAttList_1 \rangle]$ for L and δ w.r.t. $\gamma, \sigma, maxLen$ such that $Q(\mathcal{M}, L^{>\gamma})$ is maximal.

Clearly, solving this problem exactly is hardly feasible in many real settings, as a huge amount (actually combinatorial in the number of constraints that can be defined over the extended set of case attributes, including behavioral features with potentially large domains) of candidate clustering models need to be evaluated. Thus, we solve the problem through a two-phase heuristics algorithm, named DACM-mine and shown in Fig. 2.

The algorithm initially extends (through function extendCaseAttributes) the cases in L with additional attributes, devoted to encode both context information (e.g., workload indicators, environmental factors) and a summary of the associated trace. Such a pre-processing step is needed to eventually extract readable deviance patterns, which would be hard to recognize looking at the very sequence of trace events, also due to the presence of complex execution schemes (involving loops, concurrency, choices).

Steps 2-11 are meant to extract the list of clustering rules (i.e. description patterns) that will constitute the backbone of the discovered DACM. Essentially, these rules are built iteratively (one after the other) through a sequential covering scheme.

Each iteration of the main loop (Steps 3-13) tries to extract the best possible clustering rule out of dataset Z (initially encoding all the cases in L) by way of function growClusteringRule, explained later on. The instances covered by the discovered rule (hopefully identifying the most interesting deviance cluster possible), are removed from Z, and are not used to induce any subsequent rule. This procedure is repeated until no further γ-interesting cluster is found (i.e. a cluster $\hat{c}$ such that $\mathscr{I}(\hat{c}) > \gamma$) —as an extreme case,

Input: Log L with associated sets $\mathcal{A}$ (resp., $\mathcal{A}_{\mathcal{E}}$) of case (resp., event) attributes, deviance indicator $\delta : L \to [0,1]$, minimal deviance $\gamma \in [0,1)$ and minimal size $\sigma \in \mathbb{N}^+$ for deviance clusters, and maximal length $maxLen \in \mathbb{N}^+$ for clustering rules (patterns).
Output: A `DACM` model for L.
Method: Perform the following steps:

```
 1 extendCaseAttributes(L, A_E);
 2 M := []; M' := []; Z := L; continue := true;
 3 while continue do
 4    φ := ∅;  // start with a dummy rule (i.e. a pattern with no constraints) covering all the instances in Z
 5    φ := growClusteringRule(φ, Z, σ, maxLen, γ);  // notice that it is ensured |cov(φ1, Z)| ≥ σ
 6    if φ = ∅ or I(cov(φ1, Z)) < γ then continue := false;
 7    else
 8       φ := prune&OptimizeRule(φ, cov(φ, Z), σ, maxLen, γ);
 9       M.append(⟨φ*, cov(φ*, Z)⟩);    Z := Z \ cov(φ*, Z);
10    end
11 end
12 M.append(⟨∅, Z⟩);  // Insert the "default" rule, along with the corresponding set of "normal" cases, into M
13 M := prune&OptimizeModel(M);
14 for each pair ⟨φ, ĉ⟩ in M do
15    α := mineEventAbstraction(A_E, ĉ, L);
16    attrList := rankCaseAttributes(A, ĉ, L);
17    M'.append(⟨φ, ĉ, α, attList⟩);
18 end
19 return M'
```

Fig. 2. Algorithm `DACM-mine`.

this will happen when all the γ-deviant cases in Z have been covered. The ordered set of rules obtained this way is stored, along with the corresponding case clusters, into a list M of (rule, cluster) pairs. Once built each new rule ϕ, we try to optimize it heuristically through function `prune&OptimizeRule`, which possibly modifies the constraints in ϕ in the order they were added. Specifically, each constraint ξ is removed if the resulting (pruned) rule ϕ' does not decrease the deviance score of the cluster. Otherwise, it tries to replace ξ with a new constraint ξ', and confirms the modification if a better cluster is obtained than that produced by ϕ. A similar optimization procedure is iteratively applied (in Step 13), to all the rules (from the oldest) in the discovered clustering model. Each explored rule modification is kept if it (locally) improves the interestingness of the entire model (measured as in Eq. 1), or (in the case of constraint removals) it simplifies the model with no quality decrease.

The second phase of the algorithm (Steps 14-19), is devoted to equip each of the discovered clusters with an event abstraction model, and a (ranked) list of distinguishing case attributes, computed with functions `mineEventAbstraction` and `rankCaseAttributes`, respectively —details on both functions are presented later on.

The rest of the section gives more details on some computation steps of `DACM-mine`.

Function `extendCaseAttributes` In order to produce an abstract and compact representation of the behavior captured by the traces (i.e. sequences of multi-dimensional events) stored in L, for each of the event attributes in $\mathcal{A}_{\mathcal{E}}$, the function extends each case c in L with two sets of attributes: $\mathcal{B} = \{\ \#[A = v] \mid A \in \mathcal{A}_{\mathcal{E}} \text{ and } v \in Dom(A)\ \}$ and $\mathcal{B}_{pairs} = \{\ \#[A = u \to v] \mid A \in \mathcal{A}_{\mathcal{E}} \text{ and } u, v \in Dom(A)\ \}$. The two kinds of attributes store as many times a specific value (resp., two specific values) of A appears (resp., appear one after the other) in the sequence. More precisely: $\#[A = v](c) = |\{i \in \{1, \ldots, len(\tau)\} | \tau = trace(c) \text{ and } A(\tau[i]) = v\}|$;

and $\#[A = u \rightarrow v](c) = |\{i \in \{1,\ldots,len(\tau)\} | \tau = trace(c) \text{ and } \tau[i] = u \text{ and } A(\tau[i+1]) = v \text{ and } i < j\}|$. By the way, similar encodings are used other process mining approaches [9].

Function `growClusteringRule`. The induction of each clustering rule is based on a greedy (top-down) iterative pattern-refinement scheme, similar to the one used in classical rule-based classification [3] and subgroup discovery [1, 8] approaches. Starting from an empty pattern ϕ_{cur} (i.e. a pattern encoding no actual constraint, and hence covering all the training instances left uncovered by previous rules), the function tries to iteratively expand ϕ_{cur} by adding one constraint per time, hence obtaining a set of candidate refined patterns of size $|\phi_{cur}| + 1$. Among all the candidate patterns (if any) fulfilling the size requirement expressed by σ, it selects the one identifying a cluster with the highest quality score, measured through the following scoring function: $\tilde{\mathscr{I}}(\hat{c}) = \mathscr{I}(\hat{c}) \cdot f(\mathscr{I}(\hat{c})) \cdot \log\left(\sum_{c \in \hat{c} \cap L^{>\gamma}} w(c, L^{>\gamma})\right)$, where $f(x) = (1 + e^{-7.5 \cdot (x - 0.5)})^{-1}$. This function, computing a smoothed (non-normalized) of the contribution of $\hat{c}$ to $Q(\mathcal{M}, L^{>\gamma})$. was found capable to yield interesting and compact models over several datasets. This expansion procedure is stopped when either *(i)* the pattern has reached the maximal size *maxLen*, or *(i)* no refinement has ben found for current pattern ϕ that both satisfies the minimal size constraint indicated by σ, and improves the quality score of ϕ. As to the kind of constraints used when expanding a pattern, for each categorical attribute A and for each $v \in Dom(A)$, we consider both the constraints $A = v$ and $A \neq v$ (unless A is binary), as well as $|Dom(A)|$ subsets of size $1, 2, \ldots, |Dom(A)|$, respectively, selected greedily according to the method in [7]. By converse, for each numerical attribute A, we consider all conditions of the form $A \leq v$ or $A > v$, such that v is one of the cut points separating two consecutive equal-size ranges of $Dom(A)$, among all those obtained by applying a dynamical binning procedure (similarly to [1]) to Z.

Function `mineEventAbstraction`. The function computes an event abstraction function for a given cluster $\hat{c}$ by mainly applying a predictive clustering procedure [2] to an ad-hoc dataset encoding a propositional view of the events in L. Specifically, for each event e, the dataset contains a tuple z_e where the original properties of $prop(e)$ (expressed over $\mathcal{A}_E$ and regarded as descriptive features) is complemented with the following target features: a boolean flag, indicating whether e concerns a case belonging to $\hat{c}$ or not (and hence acts as a sort of binary class label), and three numeric indicators that encode the relative/absolute position occupied by e in its surrounding trace. Such an encoding scheme, similar to the one proposed in [4], is meant to help put together events with similar intra-trace positions (and similar properties), especially when they concern cases of the same class (i.e. either inside $\hat{c}$ or outside $\hat{c}$).

Function `rankCaseAttributes`. To extract a ranked list of characterizing case attributes, for each discovered cluster $\hat{c}$ and each $A \in \mathcal{A}$, function `rankCaseAttributes` compares the (sample) distribution of $\mathcal{A}$ over $\hat{c}$ with the (sample) distribution of A all over L. To this end, it exploits two popular divergence measures and associated tests, in order to rank and select the attributes: the two-sample K-S measure for numerical attributes, and KL (Kullback-Leibler) divergence for categorical ones —a case attribute is kept in the list if the two distributions look significantly (level .95) different.

5 Experiments

We assessed the capability of our approach to recognize valuable and interpretable deviance-aware clustering models, by conducting some tests on a log, named `BPIC13`, coming from a real-life problem management system —provided by Volvo IT Belgium as a benchmark

dataset for the 2013 BPI Challenge [10]. Precisely, we used 1487 traces spanning from January 2006 to May 2012, cumulatively gathering 9634 log events.

Each event stored for a (problem resolution) case p consists of 8 attributes: `status`, `substatus`, `support_team`, `functional_division`, `org_line`, and `org_country`. The `resource` currently operating on p and its nationality (`res_country`) are also registered in each event. We enriched the set of event attributes with a further one (`lev`) to represent the expertise level (namely, 1st, 2nd, and 3rd) of the involved resolution team.

We also extended each case p in the log with the following kinds of context data (encoded as case attributes): *(i)* the total number of cases open in the system (`workload`) at the time, say t, when p started; *(ii)* several time dimensions (namely, `week-day`, `month` and `year`) derived from t; and *(iii)* a few properties of the first event of p, including the support team and resource allocated to the case, and their respective countries (resp., `firstOrg`, `firstRes`, `firstOrgCountry`, and `firstResCountry`).

Each case p was associated with two cost measures: *(i)* the nr. of distinct resources that worked on p (*#Resources*), and *(ii)* the total time spent to handle p (*CycleTime*). Then, for each of these cost measures, say m, we computed the deviance indicator $\delta(p)$ of any case p as the difference between $m(p)$ and the minimum value of m in the log, normalized by the maximal difference in the log. Observe that, when analyzing deviances w.r.t. *#Resources*, we discarded the attributes `status` and `substatus`, which convey information (clearly related to the cost measure) on resource assignment.

Evaluation measures. In the test we evaluated the quality of any discovered clustering model according to two dimensions: *(i)* interestingness, and *(ii)* easiness of explanation.

For the former, we used an *Interestingness measure* (I), defined as a normalized version of the function Q of Eq. 1 —computed by dividing the value of Q by the score the latter would assign to an ideal (but overly detailed) model putting each γ-deviant case into a separate deviance cluster, and all the "normal" ones in the last cluster.

In order to evaluate the second dimension onto any clustering model $\mathcal{M}$, we introduced a new measure, named *Explanation Complexity* (EC), defined as a weighted sum of the "explanation costs" of all the deviant cases $L^{>\gamma}$: $EC(\hat{C},\gamma,L) = [\sum_{c\in L^{>\gamma}} w(c,L,\gamma) \cdot clusterID_{\mathcal{M}}(c)]/[\sum_{c\in L^{>\gamma}} w(c,L^{>\gamma})]$ where, for each γ-deviant case c, $clusterID_{\mathcal{M}}(c)$ is the number of clustering rules that one need to read to eventually interpret c, and the rank-oriented weight $w(c,L^{>\gamma})$ of c is used to weight the explanation cost of c (remember, the lower the rank, the higher weight). Notice that this measure is inspired to the *prediction-explanation size* that was defined in [5] (based on the number of conditions evaluated for eventually classifying, and explaining, a test instance) for decision-list classifiers, as a more realistic measure of interpretability than the mere list length.

Since our main goal is to find maximally deviant scenarios, the *Interestingness* measure is always prevalent over the *Explanation Complexity* one. Then, only when two approaches achieve the same (or comparable) interestingness degrees, the model compactness becomes discriminant.

5.1 Test results

Quantitative Results. Some interesting observations can be deduced from Table 1, which also reports, as a term of comparison, the results obtained with two existing methods, which both build a list of decision rules (alike those appearing in our models): the conceptual clustering method *M5Rules* [6] and the rule-based classifier *Ripper* [3]. For both of them, we used the implementation available in the popular *Weka* library, with the default parameters'

Table 1. Quality measures on the BPIC13 log by DACM-mine and the baseline methods ([6], [3]) at varying of both the performance measures and the value $x\%$ (i.e. the percentages of deviant instances allowed in the log used for setting the parameter γ). Best outcomes are shown in bold.

Setting		Interestingness (*I*)			Explanation Complexity (*EC*)		
Cost measure	*x%*	DACM-mine	Ripper [3]	M5Rules [6]	DACM-mine	Ripper [3]	M5Rules [6]
#Resources	5%	**0.935**	0.917	0.872	**1.57**	2.15	13.94
	10%	**0.955**	0.902	0.887	3.68	**3.18**	12.62
	20%	**0.955**	0.902	0.887	3.68	**3.18**	12.62
CycleTime	5%	**0.982**	0.892	0.887	2.33	**1.06**	16.85
	10%	**0.988**	0.888	0.927	2.96	**1.03**	15.45
	20%	**0.988**	0.910	0.951	3.97	**1.18**	14.14

setting. Since none of these two methods specifically addresses the problem faced in this work, we will consider them both as a sort of baseline.

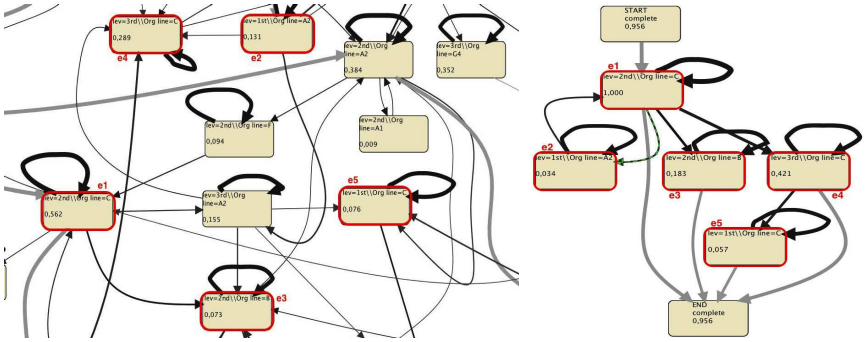
First, algorithm DACM-mine, irrespective of the cost measure and the setting adopted, always performs better than the baselines over the main *Interestingness* measure. This confirms the capability of our approach to better recognize highly deviant scenarios than less tailored techniques. Indeed, sensible average gains (over all percentages $x\%$) are obtained in terms of *Interestingness* w.r.t. *Ripper* (resp. *M5Rules*) —namely $+4.6\%$ (resp. +7.5%) when deviances are measured over the cost measure *#Resources*, and $+10\%$ (resp. $+7.1\%$) when the reference cost measure is *CycleTime*, respectively.

As to the secondary *Explanation Complexity* measure, more mixed results are shown for DACM-mine in Table 1. Indeed, it regularly overcomes algorithm *M5Rules* in terms of model compactness, but it shows the tendency to generate longer decision lists (apart from the case when the cost measure is *#Resources* and $x\% = 5\%$) than algorithm *Ripper* instead. However, it is worth noticing that *Ripper* generally tends to produce fewer rules since it performs an easier task (i.e. discriminating deviant from normal cases) than that of separating the deviant cases in groups on the base of their deviance values (as both algorithms DACM-mine and *M5Rules* do). Nonetheless, when the cost measure is *#Resources*, DACM-mine even achieves a better result in terms of *EC* (1.57) than Ripper (2.15) in the case $x\% = 5\%$, and it continues to show comparable outcomes (3.68) w.r.t. Ripper (3.18) in both the remaining cases $x\% = 10\%$ and $x\% = 20\%$, respectively.

When the deviance indicator is computed on the *CycleTime* instead, Ripper leads on *EC* by the widest margin over DACM-mine (1.09 for the former and 3.09 for the latter when averaged over $x\%$), but it pays a substantial tribute to DACM-mine in terms of *Interestingness* (in average 0.896 for the former and 0.986 for the latter) evidencing its inability to extract as interesting models as those discovered by DACM-mine.

Qualitative Results. For the sake of space, we only show here some findings obtained by DACM-mine on the log BPIC13 when using *#Resources* as cost measure and setting $x\% = 10\%$. As a result, 9 deviant clusters were found, and to give an hint, the details relative to the first four clusters are extensively reported in Fig. 4(a). Notice that these specific patterns feature only a series of attributes extracted from the cases' traces (by the function extendCaseAttributes of algorithm DACM-mine). For instance, the pattern of *cluster*#2 (covering 20 cases), shown in Fig. 4(a), is composed by three conjuncts, the first (resp., third) of which states that at least 9 (resp. at most 18) events concern the organizational line C (resp. the 2nd level), while the second conjunct states that at least a couple of events associated with the 2nd level occur consecutively.

In addition to the cluster description, each cluster is also equipped with a separate workflow schema meant to visually summarize the deviant behavior captured by the cluster. Figure 3 (b) shows a process map derived (using the plug-in *Fuzzy Miner* available in



(a) (b)

Fig. 3. Process maps discovered, w.e.t. the cost measure #*Resources*, for the whole log (a) and the deviance *cluster*#2 (b). Nodes corresponds to event patterns that were discovered for the cluster. E.g., node e_1 corresponds to the pattern "`lev` = 2nd $\wedge$ `org_line` = C'.

ProM [11]) for *cluster*#2, side-by-side to an excerpt of the global map derived from the whole log (Fig. 3 (a)). Notably, both maps were produced by preliminary making each log event undergo the event abstraction model automatically discovered by `DACM-mine` for *cluster*#2 — which, for this specific cluster, essentially projects the events onto the sole attributes `lev` and `org_line`. Besides being more compact than the global process map, the map in Fig. 3 (b) helps us reckon that the 20 problems assigned to *cluster*#2 follow quite a particular handling policy. Indeed, (part of) the deviant cases followed a path toward their completion (i.e. $e_1 \rightarrow e_4 \rightarrow e_5$) that is not so frequent in the whole log. The same holds for another feature captured by the map of *cluster*#2, where the cycle $e_1 \rightarrow e_2 \rightarrow e_1$ is possibly executed multiple times before completion.

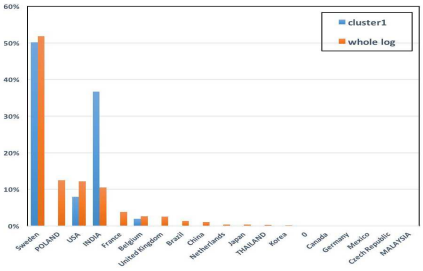
Finally, Fig. 4 (b) reports a chart where the distribution within *cluster*#2 of case attribute `firstResCountry` (reckoned by `DACM-mine` as a highly characterizing one for the cluster) is contrasted to the global distribution of the same attribute.

6 Discussion and Future work

The framework proposed above is original in several respect, compared to current approaches to the explanation of quantitative business process deviances. Indeed, to the best of our knowledge there is no existing method that can solve effectively the specific problem stated in this work, where *(i)* a number (not known a-priori) of deviance scenarios are searched for, which may exhibit quite different levels of deviance, and *(ii)* and an interpretable explanation is to be provided for each of them (in terms of either context factors or behavioral features, or both). Experimental findings confirm that our approach overcomes the naive solution of employing an existing rule-based classifier or conceptual method to solve the problem. As an alternative, one could think of exploiting solutions developed in the fields of Subgroup Discovery and of Exceptional Model Mining. However, most of these are likely not not fit well our setting, due to their inherent risk of producing two many (overlapping) subgroups of cases, which exhibit lowly different descriptions and hence capture highly correlated deviance scenarios. Anyway, as future work, we plan to investigate on possibly hybridizing our approach with such techniques, while trying to make them fit better our problem setting —to this respect, we will pay attention to recent efforts [8] towards producing an optimal collection of subgroups having a minimal global degree of overlap.

cluster id	cluster description (pattern)	cluster size
1	#[lev = 3rd] > 0 ∧ #[org_line = C] > 11	19
2	#[org_line = C] > 9 ∧ #[lev = 2nd → 2nd] > 0 ∧ #[lev = 2nd] ≤ 18	20
3	#[org_line = A2 → G3] > 0 ∧ #[org_line = A2] > 2 ∧ #[lev = 3rd] > 6	21
4	#[org_line = C → B] > 0	10
…	…	…

(a)



(b)

Fig. 4. Decision list (for 4 patterns out of 9) induced by `DACM-mine` for the log `BPIC13` (a), and distribution of attribute `fistResCountry` in the *cluster*#2 compared to that in the whole log (b).

Moreover, we will try to improve both the scalability of our learning method (e.g., by exploiting a parallelized computation of clusters and fast indexing structures) and the quality of the mined clusters (e.g., by replacing our greedy sequential-covering scheme with a beam search one).

References

1. Atzmueller, M.: Subgroup discovery - advanced review. Wiley Int. Rev. Data Min. and Knowl. Disc. 5(1), 35–49 (Jan 2015)
2. Blockeel, H., Raedt, L.D., Ramon, J.: Top-down induction of clustering trees. In: Proc. of the 15th Int. Conf. on Machine Learning (ICML'98). pp. 55–63 (1998)
3. Cohen, W.W.: Fast effective rule induction. In: Proc. of 12th Int. Conf. on Machine Learning (ICML'95). pp. 115–123 (1995)
4. Folino, F., Guarascio, M., Pontieri, L.: Mining multi-variant process models from low-level logs. In: Proc. of 18th Int. Conf. on Business Information Systems (BIS'15). pp. 165–177 (2015)
5. Freitas, A.A.: Comprehensible classification models: A position paper. SIGKDD Explor. Newsl. 15(1), 1–10 (2014)
6. Holmes, G., Hall, M., Frank, E.: Generating rule sets from model trees. In: Proc. of 12th Australian Joint Conf. on Artificial Intelligence (AI'99). pp. 1–12 (1999)
7. L.Breiman, Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and RegressionTrees. Wadsworth, Belmont (1984)
8. Leeuwen, M., Knobbe, A.: Diverse subgroup set discovery. Data Min. Knowl. Discov. 25(2), 208–242 (2012)
9. Leontjeva, A., Conforti, R., Francescomarino, C.D., Dumas, M., Maggi, F.M.: Complex symbolic sequence encodings for predictive monitoring of business processes. In: Proc of 13th Int. Conf. on Business Process Management (BPM'15). pp. 297–313 (2015)
10. Steeman, W.: BPI challenge 2013, closed problems. doi:10.4121/c2c3b154-ab26-4b31-a0e8-8f2350ddac11 (2013)
11. Verbeek, H.M.W., Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: Xes, xesame, and prom 6. In: Information Systems Evolution - CAiSE Forum 2010, Selected Extended Papers. pp. 60–75 (2010)