On the Discovery of Explainable and Accurate Behavioral Models for Complex Lowly-structured Business Processes

Francesco Folino, Massimo Guarascio and Luigi Pontieri
Institute ICAR, National Research Council (CNR), via P. Bucci 41C, 87036, Rende, CS, Italy

Keywords: Data Mining, Business Process Intelligence, Trace Clustering, Workflow Discovery.

Abstract: Process discovery (i.e. the automated induction of a behavioral process model from execution logs) is an important tool for business process analysts/managers, who can exploit the extracted knowledge in key process improvement and (re-)design tasks. Unfortunately, when directly applied to the logs of complex and/or lowly-structured processes, such techniques tend to produce low-quality workflow schemas, featuring both poor readability ("spaghetti-like") and low fitness (i.e. low ability to reproduce log traces). Trace clustering methods alleviate this problem, by helping detect different execution scenarios, for which simpler and more fitting workflow schemas can be eventually discovered. However, most of these methods just focus on the sequence of activities performed in each log trace, without fully exploiting all non-structural data (such as cases' data and environmental variables) available in many real logs, which might well help discover more meaningful (context-related) process variants. In order to overcome these limitations, we propose a two-phase clustering-based process discovery approach, where the clusters are inherently defined through logical decision rules over context data, ensuring a satisfactory trade-off is between the readability/explainability of the discovered clusters, and the behavioral fitness of the workflow schemas eventually extracted from them. The approach has been implemented in a system prototype, which supports the discovery, evaluation and reuse of such multi-variant process models. Experimental results on a real-life log confirmed the capability of our approach to achieve compelling performances w.r.t. state-of-the-art clustering ones, in terms of both fitness and explainability.

1 INTRODUCTION

Process discovery (more precisely, control-flow discovery) techniques (van der Aalst et al., 2003) are a valuable tool for automatically extracting a behavioral schema for a business process (out of past execution traces), which can profitably support key process analysis, (re-)design, and optimization tasks.

Unfortunately, a direct application of such techniques to the logs of lowly-structured processes (featuring a large variety of behavioral patterns) is likely to yield low quality ("spaghetti-like") workflow schemas, exhibiting both low readability and low fitness (Buijs et al., 2012). By contrast, reliable workflow schemas would be very important for many real-life flexible and dynamic process management settings, where little a-priori knowledge is available on what typical work schemes are followed in reality.

To alleviate such a problem, several trace clustering approaches (De Weerdt et al., 2013; Bose and van der Aalst, 2009a; Bose and van der Aalst, 2009b; Song et al., 2008; Greco et al., 2006) have been proposed in the literature, which help recognize different homogenous execution scenarios (or "process variants"), each of which can be effectively described through a simpler and more fitting workflow schema.

However, most of these approaches only focus on structural aspects of the traces (e.g., which activities were performed, and in what an order), paying no attention at all to all kinds of non-structural data (such as qualitative/quantitative properties of process cases, or other context factors characterizing the state of the execution environment at the moment when the case was performed), which are often available in most real logs. And yet, such data may well help the analyst get insight on the discovered execution scenarios, whenever the behavior of the process tends to be correlated to non-structural context variables.

In fact, it was shown in (Folino et al., 2008; Folino et al., 2011) that such correlations between non-structural variables and process behaviors do exist in some real application scenarios, and it is possi-

Folino F., Guarascio M. and Pontieri L..
On the Discovery of Explainable and Accurate Behavioral Models for Complex Lowly-structured Business Processes.
DOI: 10.5220/0005470602060217
In *Proceedings of the 17th International Conference on Enterprise Information Systems* (ICEIS-2015), pages 206-217
ISBN: 978-989-758-096-3

ble to learn a classification model for discriminating among different behavioral classes discovered with a trace clustering procedure. However, the application of standard classifier-induction methods, as a post-processing step, to the results of a purely-structural trace clustering method is not guaranteed to achieve good accuracy performances in general — as confirmed by experimental findings presented in Section 6.

To overcome the above limitations we here try to face a new kind of clustering-oriented process discovery problem, specifically tailored to the case of complex and/or lowly-structured process logs. The ultimate goal of our approach is to discover a high-quality multi-variant process model for a given log, consisting of different workflow schemas, one for each of discovered trace clusters, which ensures an optimal trade-off between: *(i)* the readability/interpretability of the discovered clusters and of their distinguishing features (possibly linked to context factors), on the one hand, and *(ii)* the behavioral fitness of the discovered workflow schemas (i.e., the capability to adequately capture the behaviors registered in the log), on the other hand. As to the former point, for a certain level of behavioral fitness, higher readability/interpretability is ensured by models featuring a lower number of clusters, easily explainable in terms of (accurate enough) discriminating rules over non-structural context variables.

In order to obtain clusters inherently correlated to non-structural data, we focus on a specific family of conceptual clustering models, represented as logical (decision) rules. Technically, the search for such a model is carried out by adopting a predictive clustering approach (Blockeel and Raedt, 1998), where context-oriented case variables are called to play as descriptive attributes, while the target of prediction is a number of simple structural patterns, capturing basic intra-trace precedence relationships between process activities. As a result, a preliminary set of clusters is found, each associated with a specific decision rule representing a specific setting of context variables (i.e. a specific "context" variant). Since there is no guarantee that the discovered clusters really correspond to neatly different behavioral schemes, a greedy iterative restructuring procedure is carried out, where redundant clusters showing similar behaviors are merged together, as long as the average fitness of the associated workflow schemas can be increased.

The whole approach has been implemented into a system prototype, which fully assists the user in discovering, inspecting and evaluating such multi-variant process models, and helps reuse them in advanced analyses and run-time support tasks. In particular, the system is meant to both help the analyst inspect and validate the correlations discovered between non-structural context factors and the structure of process instances (encoded in the form of logical clustering rules), as well as to exploit them for the provision of advanced run-time services, which can be very useful when enacting flexible and dynamic processes. In particular, by applying the discovered classification model to a partially unfolded (i.e. not finished) process case, it is possible to show the workflow schema of the cluster it is estimated to belong to, as a customized (context-adaptive) process map, describing how the case may proceed.

Experimental results confirmed its capability to achieve compelling performances, in terms of both fitness and explainability, with respect to several state-of-the-art clustering approaches: algorithm `Actitrac` (De Weerdt et al., 2013); the sequence-based and alphabet-based versions of the approach proposed in (Bose and van der Aalst, 2009b), based on the mapping of log traces onto a space of behavioral patterns (namely, *tandem repeats* and *maximal repeats*, respectively); the approach proposed in (Bose and van der Aalst, 2009a), which exploits a *k*-gram representation of traces; and `DWS` algorithm (Greco et al., 2006), which recursively partitions a log based on a sequential patterns capturing unexpected behaviors.

The rest of the paper is organized as follows. We first present some related research work in Section 2, and a few basic concepts in Section 3. The core technical framework is described in Section 4, while Section 5 illustrates our discovery approach and system prototype. An empirical evaluation of our proposal on a real-life log is discussed in Section 6, while a few concluding remarks are drawn in Section 7.

2 RELATED WORK

Several clustering approaches have been proposed in the literature, which try to exploit different kinds of information captured in log traces, in order to help recognize different behavioral classes of process instances automatically.

Some of these solutions leverage sequence-oriented techniques (Ferreira et al., 2007; Bose and van der Aalst, 2009a), which compare entire traces by way of string-oriented distance measures. For instance, the context-based approach defined in (Bose and van der Aalst, 2009a) relies on the generic edit distance, embedded within an agglomerative clustering scheme. A probabilistic approach was proposed instead in (Ferreira et al., 2007), where a mixture of

first-order Markov models approximating the distribution of log traces (still considered as sequences) is computed via an EM scheme.

Unfortunately, all of these string-oriented techniques require very expensive computations, which may make them unsuitable for the analysis of massive logs. Conversely, higher scalability can be achieved by resorting to feature-based approaches, such as those in (Greco et al., 2006; Song et al., 2008; Bose and van der Aalst, 2009b), which reuse efficient clustering methods defined for vectorial data, after projecting each trace onto some space of derived features.

In more detail, different kinds of features were considered in (Song et al., 2008) to this purpose, as a way to capture the behavior of a trace according to different perspectives (activities, transitions, data, performance values, etc). Each of such features is associated with a measure that assigns a numeric value to the feature over any possible trace. After replacing each trace with a vector storing such measures, whatever distance-based clustering method can be reused to partition the log. In particular, in (Song et al., 2008), the usage of three standard distance measures (namely, the Euclidean distance, Hamming distance and Jaccard coefficient) was investigated, in combination with four alternative clustering schemes (namely, K-means, Quality Threshold Clustering, Agglomerative Hierarchical Clustering and Self-Organizing Maps).

As a more expressive kind of trace features, it was proposed in (Bose and van der Aalst, 2009b) to exploit certain sequential patterns inspired to bioinformatics (including *tandem arrays* and *tandem repeats*), which allow for capturing recurring groups of correlated activities and loop structures. After extracting a set of frequent patterns of such a form from all the given log, each trace can be transformed into a vector storing how many times each of them occurs in the trace. In order to effectively deal with the presence of concurrent behavior, a variant (named "alphabet-based") of this vector-like encoding was also defined in (Bose and van der Aalst, 2009b), based on the very idea of regarding all patterns sharing the same set of activities (i.e. defined over the same "alphabet") as just one dimension of the target space. In this way, two distinct sequential patterns, e.g. the repeats *abdgh* and *adgbh*, will be viewed as a unique feature.

It is worth noticing that, in fact, (Bose and van der Aalst, 2009b) also explored the possibility to convert all log traces into the (abstracted) sequences of patterns occurring in them (acting as a sort of typical sub-processes), and then comparing them by way of some edit-distance measures.

A top-down recursive clustering scheme was proposed in (Greco et al., 2006), where, at each step, a workflow model is extracted from a set of log traces, which may be further partitioned (until a maximal number of clusters is reached) in order to obtain a collection of more precise models (representing distinguished execution scenarios). In order to accomplish each of these trace clustering tasks, algorithm k-means is applied to an ad-hoc propositional representation of the traces, based on sequential patterns (named "discriminant rules") capturing unexpected behaviors (w.r.t. the workflow schema currently associated with the traces that are to be partitioned).

A fitness-aware trace clustering approach was finally proposed in (De Weerdt et al., 2013), which tries to group the traces in a way that a user-specified level of fitness for each output process model is achieved.

Differently from our work, all the above clustering approaches are not concerned with the problem of discovering a partition of the given traces that is strongly correlated with context-related factors. Such a problem was partly addressed in (Folino et al., 2008; Folino et al., 2011), where a classification model was learnt from a set of trace clusters (discovered by way of the structural clustering method in (Greco et al., 2006), in order to possibly discriminate among them, on the basis of non-structural information associated with the traces. However, the application of standard classifier-induction methods, as a post-processing step, to the results of a purely-structural trace clustering method is not guaranteed to achieve good accuracy performances. As mentioned in the previous section, this is the reason why we prefer to adopt a predictive clustering approach, focusing only on partitions of the traces that can be defined in terms of logical rules over their associated non-structural (context-oriented) data.

Before concluding this section, let us notice that the usage of predictive clustering techniques (Blockeel and Raedt, 1998) is not completely novel in a process mining setting. Indeed, it was originally proposed in (Folino et al., 2012), but with the different aim of supporting run-time predictions for a case-oriented performance metrics. In fact, the basic idea of predictive clustering is that, once discovered a suitable clustering, accurate predictions for new instances can be made by first estimating the clusters they are deemed to belong to.

3 PRELIMINARIES

Log Traces. For each process instance (a.k.a "case") we assume that a *trace* is recorded, storing the sequence of *events* happened during its enact-

ment. Let $\mathcal{T}$ be the universe of all (possibly partial) traces that may appear in any log of the process under analysis. For any trace $\tau \in \mathcal{T}$, $len(\tau)$ is the number of events in τ, while $\tau[i]$ is the i-th event of τ, for $i = 1$ .. $len(\tau)$, with $task(\tau[i])$ and $time(\tau[i])$ denoting the task and timestamp of $\tau[i]$, respectively. We also assume that the first event of each trace is always associated with a unique "initial" task (possibly added artificially), and its timestamp registers the time when the corresponding process instance started.

For any trace τ, let $context(\tau)$ be a tuple gathering a series of data about the execution context of τ, ranging from intrinsic data properties to environmental variable characterizing the state of the BPM system when τ was enacted.

For ease of notation, let $\mathcal{A}^{\mathcal{T}}$ denote the set of all the tasks (a.k.a., activities) that may occur in some trace of $\mathcal{T}$, and $context(\mathcal{T})$ be the space of context vectors — i.e., $\mathcal{A}^{\mathcal{T}} = \cup_{\tau \in \mathcal{T}} tasks(\tau)$, and $context(\mathcal{T}) = \{context(\tau) \mid \tau \in \mathcal{T}\}$.

Finally, a *log* L is a finite subset of $\mathcal{T}$.

Workflow Schemas and Behavioral Profiles. Various languages have been proposed in the literature for specifying the behavior of a business process, in terms of its composing activities and their mutual dependencies — such as *Petri nets* (van der Aalst, 1998), *causal nets* (van Der Aalst et al., 2011), and *heuristics nets* (Weijters and van der Aalst, 2003). For the sake of concreteness we next focus on the language of *heuristics nets* (Weijters and Ribeiro, 2011), where a *workflow schema* is a directed graph where each node represents a process activity, each edge (x,y) encodes a dependency of y on x, while each fork (resp., join) node can be associated with cardinality constraints over the the edges exiting from (resp., entering) it.

The behavior modeled by a workflow schema can be captured approximately by way of simple pairwise relationships between the activities featuring in it, named *(causal) behavioral profiles* (Weidlich et al., 2011), which can be computed efficiently for many classes of models.

Let W be a workflow schema, and $\mathcal{A}(W)$ be its associated activities. Let $\succ^W$ be a "weak order" relation inferred from W, such that, for any $x,y \in \mathcal{A}(W)$, it is $y \succ^W x$ iff there is at least a trace admitted by W where y occurs after x. Then the *behavioral profile matrix* of W, denoted by $\mathcal{B}(W)$, is a function mapping each pair $(x,y) \in \mathcal{A}(W) \times \mathcal{A}(W)$ to an ordering relation in $\{\rightsquigarrow, +, \|\}$, as follows: **(i)** $\mathcal{B}(W)[x,y] = \rightsquigarrow$, iff $y \succ^W x$ and $x \not\succ^W y$ (*strict order*); **(ii)** $\mathcal{B}(W)[x,y] = +$, iff $x \not\succ^W y$ and $y \not\succ^W x$ (*exclusiveness*); **(iii)** $\mathcal{B}(W)[x,y] = \|$, iff $x \succ^W y$ and $y \succ^W x$ (either *interleaving* or *loop*).

Let τ be a trace, over trace universe $\mathcal{T}$, x and y be two activities in $\mathcal{A}^{\mathcal{T}}$, and $\mathcal{B}$ be a behavioral profile matrix. Then we say that τ *violates* (resp., *satisfies*) $\mathcal{B}[x,y]$, denoted by $\tau \nvdash \mathcal{B}[x,y]$ (resp., $\tau \vdash \mathcal{B}[x,y]$), if the occurrences of x and y in τ infringe (resp., fullfill) the ordering constraint stated in $\mathcal{B}[x,y]$. More specifically, it is $\tau \nvdash \mathcal{B}[x,y]$ iff there exist $i, j \in \{1,...,len(\tau)\}$ such that $\tau[i] = y$, $\tau[j] = x$, and either *(i)* $\mathcal{B}[x,y] = +$, or *(ii)* $\mathcal{B}[x,y] = \rightsquigarrow$ and $i < j$.

Conceptual Clustering Models. The core assumption under our work is that the behavior of a process depends on context factors. Hence, in order to predict the structure of any trace τ, we regard its associated context properties $context(\tau)$ as descriptive attributes.

For the sake of interpretability, we seek a conceptual clustering model encoded in terms of decision rules over $context(\mathcal{T})$. Let us define *conceptual clustering rule* over a trace universe $\mathcal{T}$ as a disjunction of conjunctive boolean formulas, of the form $[(A^1_1 \in V^1_1) \wedge (A^1_2 \in V^1_2) \wedge \ldots \wedge (A^1_{k_1} \in V^1_{k_1})] \vee [(A^2_1 \in V^2_1) \wedge \ldots \wedge (A^2_{k_2} \in V^2_{k_2})] \vee \ldots \vee [(A^n_1 \in V^n_1) \wedge \ldots \wedge (A^n_{k_n} \in V^n_{k_n})]$, where $n, k_1, \ldots, k_n \in \mathbb{N}$, and, for each $i \in \{1,\ldots,n\}$ and $j \in \{1,\ldots,k_i\}$, A^i_j is a descriptive attribute defined on $\mathcal{T}$'s instances (i.e. one of the dimensions of the space $context(\mathcal{T})$), and V^i_j is a subset of the domain of attribute A^i_j.

For any $L \subseteq \mathcal{T}$ and for any such a rule r, let $cov(r,L)$ be the set of all L's traces that satisfy r.

A *conceptual clustering* for L is a pair $\mathcal{C} = \langle CS, \mathcal{R} \rangle$, such that $CS = \{c_1,...,c_n\}$ is a partition of L into n clusters (for some $n \in \mathbb{N}$), and $\mathcal{R}$ is a function mapping the clusters in CS to mutually-exclusive rules like those above —note that $\bigcup_{i=1}^n c_i = L$ and $\bigcap_{i=1}^n c_i = \emptyset$), while $cov(\mathcal{R}(c), L) = c$ for any c in CS. Due to their generality, such rules can split any subset L' of Z into n clusters — $\{cov(\mathcal{R}(c_1), L'), \ldots, cov(\mathcal{R}(c_n), L')\}$ is indeed a partition of L'.

In this work, we propose to discover a special kind of conceptual clustering model for a given set of log traces, by resorting to a predictive clustering approach (Blockeel and Raedt, 1998). In general, in a predictive clustering setting, two kinds of attributes are assumed to be available for each element z of a given space $Z = X \times Y$ of instances: *descriptive* attributes and *target* ones, denoted by $descr(z) \in X$ and $targ(z) \in Y$, respectively. Hence, the goal is to find a partitioning function (similar to the conceptual clustering models above) that minimizes $\sum_{C_i} |C_i| \times Var(\{targ(z) \mid z \in C_i\})$, where variable C_i ranges over current clusters, and $Var(S)$ is the variance of set S. In our setting, the context data associated with each trace will be used as its descriptive features, whereas some basic behavioral patterns ex-

tracted from its structure (i.e. from the sequence of activities appearing in it) will be used as targets.

4 FORMAL FRAMEWORK

The kind of process model we want to eventually find mixes a collection of workflow schemas, associated with distinct execution clusters (regarded as alternative execution scenarios), with a conceptual trace clustering model (i.e. a collection of decision rules) that allows for discriminating among the clusters, based on their link to relevant context factors. Such a model is formally defined next.

Definition 1 (MVPM Model)**.** *Given a workflow log L, a multi-variant process model (short MVPM) for L is triple $M = \langle CS, \mathcal{W}, \mathcal{C} \rangle$ such that:*

- *$CS = \{L_1, \ldots L_n\}$ is a set of trace clusters, for some $n \in \mathbb{N}$, defining a partition of L —i.e., $\bigcap_{i=1..n} L_i = \emptyset$ and $\bigcup_{i=1..n} L_i = L$;*
- *$\mathcal{W}$ is a function mapping each cluster $c \in CS$ to a workflow schema, denoted by $\mathcal{W}(c)$;*
- *$\mathcal{R}$ is function mapping each cluster $c \in CS$ to a conceptual clustering rule, denoted by $\mathcal{R}(c)$, such that $\mathcal{C}_M = \langle CS, \{\mathcal{R}(c) | c \in CS\} \rangle$ is a conceptual clustering for $\mathcal{T}$.*

The size *of $\mathcal{M}$, denoted by $size(M)$, is the number of clusters in CS with it — i.e. $size(M) = |CS|$.*

In this way, L's traces are split into different (behaviorally homogenous) clusters, each of which is discriminated by a specific clustering rule (boolean formula over traces' properties), associated with the cluster by function $\mathcal{R}$. Each of these clusters, viewed as a distinguished process variant, is also associated with a workflow schema (via function $\mathcal{W}$), summarizing how process activities are typically executed in that cluster.

Problem Statement and Solution Strategy. Conceptually, the induction problem faced in this paper may be stated as the search for a MVPM of minimum size among those maximizing some conformance measure, quantifying the ability of the model to describe the behaviors registered in the input log (or in a different test log used for validation).

Various log conformance metrics have been proposed in the literature (see, e.g., (Alves de Medeiros et al., 2008; Rozinat and van der Aalst, 2008)) to compare the behavior registered in the log to that modeled by the schema. Since most of these metrics (usually defined for Petri net models) rely on a log replay strategy and imply expensive model states' explorations, they cannot be integrated in our search for an optimal MVPM. Therefore, we next introduce an approximated (but scalable) conformance measure, named fit_{BP}, which simply compares the behavioral profiles of each schema appearing in the MVPM with the corresponding sub-log it was discovered from. Given a MVPM M and a log L, the fitness of M w.r.t. L, denoted by $fit_{BP}(M,L)$, is defined as follows:

$$fit_{BP}(M,L) = \sum_{L_i \in M.CS} |L_i| \times \lambda(\mathcal{W}(L_i), L_i) \quad (1)$$

where, for any trace cluster L_i and workflow schema W_i, $\lambda(W_i, L_i) = \frac{1}{|\mathcal{A}(W_i)|^2} \times |\{(x,y) \in \mathcal{A}(W_i) \times \mathcal{A}(W_i) \mid \neg\exists\ \tau \in L_i \text{ s.t. } \tau \not\models \mathcal{B}(W_i)[x,y]\}|$. Notice that the function λ quantifies the fraction of W_i's behavioral profiles that are not violated by L_i's traces, used here as a rough (but scalable) fitness score.

To solve such a discovery problem efficiently, we devised a two-phase strategy, consisting of two main computation tasks: *(i)* extract a (possibly fine-grain) conceptual clustering CM for L, by using a simplified propositional representation of the traces, where the structure of each of them (i.e. the sequence of activities featuring in it) is encoded into a vector of basic behavioral features; *(ii)* restructure CM, by merging together as many clusters as possible, provided that the actual fitness of the workflow schemas associated with CM's clusters does not decrease.

Notably, for scalability reasons, the actual quality (measured through function fit) of the workflow associated with each trace cluster is totally disregarded in the first phase, where the search of a clustering solution only tries to minimize the expected information loss over an approximated flat representation of the traces, according to a predictive clustering approach (Blockeel and Raedt, 1998). In particular, to accomplish the first task, we resort to an existing PCT learning method (CLUS), provided with an ad-hoc encoding of the log, named *p-view* and described in details in what follows.

Propositional Trace Encoding (used in Phase I). Basically, we want to use the context data of the traces (and possibly the activities occurring in them) as descriptive features for partitioning the given log into behavioral clusters, by way of suitable clustering rules. To concisely represent major behavioral aspects of the traces, a target variable is defined over each activity pair, capture basic precedence relationships.

In more details, let $\mathcal{B}$ be the given behavioral profile matrix of some preliminary workflow schema, trying to capture all possible behaviors of the process analyzed. Then, for each trace

τ and each pair (a_i, a_j) of activities, we can define a target variable $v(\tau, a_i, a_j)$ as follows: **(i)** $v(\tau, a_i, a_j) = \frac{f(\tau, a_i, a_j)}{2 \times c(\tau, a_i, a_j)}$ if both a_i and a_j occur in τ, where $f(\tau, a_i, a_j) = \texttt{sum}(\{\texttt{sgn}(j' - i') \mid i', j' \in \{1, \ldots, len(\tau)\} \wedge task(\tau[i']) = a_i \wedge task(\tau[j']) = a_j\})$ —with sgn denoting the signum function— and $c(\tau, a_i, a_j) = |\{(i', j') \mid i', j' \in \{1, \ldots, len(\tau)\} \wedge i' \neq j' \wedge task(\tau[i']) = a_i \wedge task(\tau[j']) = a_j\}|$; and **(ii)** $v(\tau, a_i, a_j) = \texttt{null}$ if does not contain both a_i and a_j. In this way, $v(\tau, a_i, a_j)$ keeps trace of the mutual positions of a_i and a_j, if both occur in τ (case *ii*); otherwise (case *ii*), we just set $v(\tau, a_i, a_j) = \texttt{null}$.

Based on such structural target variables, we next formally define the propositional view (*p-view*) of a log, to be eventually used to extract a conceptual clustering model, by way of a PCT learner.

Definition 2 (Log View)**.** *Let L be a log over trace universe $\mathcal{T}$, and $A^{\mathcal{T}} = \{a_1, ..., a_k\}$ be the associated activity set. Then, the propositional view (short* p-view*) of L, denoted by $\mathcal{V}(L)$, is a relation containing a tuple $z_\tau = descr(z_\tau) \oplus targ(z_\tau)$ (with $\oplus$ denoting tuple concatenation) for each $\tau \in L$, such that:* ***(i)*** *$descr(z_\tau) = context(\tau) \oplus TV(\tau)$, where $TV(\tau)$ is a vector in $\{0,1\}^k$ s.t., for each $i \in \{1, .., k\}$, $TV(\tau)[i]$=1 iff a_i occurs in τ; and* ***(ii)*** *$targ(z_\tau) = \langle v(\tau, a_1, a_1), ..., v(\tau, a_1, a_k), v(\tau, a_2, a_2), ..., v(\tau, a_2, a_k), ..., v(\tau, a_2, a_k), ..., v(\tau, a_{k-1}, a_{k-1}), ..., v(\tau, a_{k-1}, a_k), ..., v(\tau, a_k, a_k)\rangle$.* □

Workflow Schema Similarity (used in Phase II) Rather than exploring all possible sequences of pairwise merges, over the clusters found in the first phase, we propose to adopt a greedy iterative agglomeration procedure, where the two clusters exhibiting the most similar behavioral models are considered for being possibly merged, at each iteration. In order to efficiently estimate how similar two workflow schemas are to each other, we next introduce an approximated similarity function, defined as a variant of that proposed in (Kunze et al., 2011).

Definition 3 (Schema Similarity)**.** *Let W_i and W_j be two workflow models, A_i, A_j be their respective activity sets (i.e. $A_i = \mathcal{A}(W_i)$, and $A_j = \mathcal{A}(W_j)$), and $\mathcal{B}_i$, $\mathcal{B}_j$ their respective behavioral profiles (i.e. $\mathcal{B}_i = \mathcal{B}(W_i)$, and $\mathcal{B}_j = \mathcal{B}(W_j)$). Let $S^h_k(W_i) = \{(x,y) \mid (x,y) \subseteq \mathcal{A}(W_i) \cap \mathcal{A}(W_j) \wedge \mathcal{B}_k[x,y] = h \wedge h \in \{\rightsquigarrow, +, \|\} \wedge k \in \{i, j\}\}$ be the ordering relationships of type h that W_i and W_j share. Then, the BP-similarity between W_i and W_j, denoted by $sim_{BP}(W_i, W_j) : W_i \times W_j \rightarrow [0,1]$ defined as:*

$$sim_{BP}(W_i, W_j) = \sum_{h \in \{\rightsquigarrow, +, \|\}} \beta_h \cdot \mathcal{J}(S^h_i, S^h_j) + \beta_A \cdot \mathcal{J}(A_i, A_j)$$

Input: A log L over trace universe $\mathcal{T}$,
minimal clusters' coverage $minCov \in (N)$.
Output: A MVPM for L;
Method: Perform the following steps:
*// **Phase I: Build an Initial Trace Clustering Model***
1. $V := \mathcal{V}(L)$; *// compute a p-view for L (cf. Def. 2)*
2. $\langle S, R\rangle := \texttt{minePC}(V, minCov)$; *//S is an L's partitioning*
// and $\{R(c) | c \in S\}$ is a conceptual clustering model
3. **for each** cluster $c \in S$
4. $\mathcal{W}(c) := \texttt{mineWFS}(c)$;
5. **end for**
*// **Phase II: Bottom-up Clusters' Merging***
6. $Q := \{\{x,y\} \mid x,y \in S\}$ *// mergeable clusters' couples*
7. **repeat**
8. **let** $(\hat{c}_1, \hat{c}_2) = \arg\max_{c_i, c_j \in Q} \{sim_{BP}(\mathcal{W}(c_i), \mathcal{W}(c_j))\}$
9. $c_{new} := \hat{c}_1 \cup \hat{c}_2$; $W_{new} := \texttt{mineWFS}(c_{new})$;
10. $Q := Q - \{\{x,y\} \mid \{x,y\} \subseteq \{\hat{c}_1, \hat{c}_2\}\}$
$\cup \{\{x, c_{new}\} \mid x \in S - \{\hat{c}_1, \hat{c}_2\}\}$;
11. **if** $|\hat{c}_1| \cdot \lambda(\mathcal{W}(\hat{c}_1), \hat{c}_1) + |\hat{c}_2| \cdot \lambda(\mathcal{W}(\hat{c}_2), \hat{c}_2)$
$\leq |c_{new}| \cdot \lambda(W_{new}, c_{new})$ **then**
12. $S := S - \{\hat{c}_1, \hat{c}_2\} \cup \{c_{new}\}$;
13. $\mathcal{W}(c_{new}) := W_{new}$;
14. $\mathcal{R}(c_{new}) := \mathcal{R}(\hat{c}_1) \vee \mathcal{R}(\hat{c}_2)$;
15. **end if**;
16. **until** $Q = \emptyset$;
17. **return** $\langle S, \mathcal{W}, \mathcal{R}\rangle$

Figure 1: Algorithm MVPM-mine.

where, for any sets X and Y, $\mathcal{J}(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$ is the Jaccard coefficient of X and Y, while $\beta_h, \beta_A \in [0,1]$ are weights such that $\sum_{h \in \{\rightsquigarrow, +, \|\}} \beta_h + \beta_A = 1$. □

In this way, the more two workflow models overlap over their activities and behavioral profiles, the more similar they are deemed. Notice that, in the experiments discussed later on, the four components of the measure above were all weighted uniformly —i.e. we set $\beta_{\rightsquigarrow} = \beta_+ = \beta_{\|} = \beta_A = 1/4$.

5 DISCOVERY APPROACH

5.1 Algorithm MVPM-mine

Figure 1 illustrates an algorithm, named MVPM-mine, for inducing a MVPM out of a given log. As mentioned previously, the algorithm follows a two-phase approach: it first finds a preliminary conceptual clustering model (Steps 1-3) for the traces, and then restructures the discovered clusters, along with their associated clustering rules and workflow schemas (Steps 4-18), via a bottom-up iterative aggregation scheme.

The first phase simply amounts to encoding the input log traces into a propositional dataset (Step 1) — where each trace is equipped with both descriptive variables and summarized behavioral features, as for-

mally defined in Def. 2— before giving it as input to function `minePC` (Step 2). This function implements the PCT-learning method described in (Blockeel and Raedt, 1998), and returns a conceptual clustering model, here denoted as a set C of trace clusters with their associated clustering rules —for each cluster c, $\mathcal{R}(c)$ is the rule that allows for discriminating c from all other clusters. Essentially, this function relies on a top-down partitioning scheme, where a split (expressed in terms of descriptive attributes) that locally minimizes clusters' variance (over the target space) is greedily selected at each step, provided that reduction of variance is significant enough (according to an F-test) and that the selected cluster contains *minCov* traces at least.

Each discovered trace cluster (capturing a context-dependent process execution variant) is then equipped with a workflow schema (Steps 3-5), by using function `mineWFS`, which implements the *Flexible Heuristics Miner (FHM)* method described in (Weijters and Ribeiro, 2011) —this choice was mainly due to the scalability and robustness to noise of this method.

It is worth noticing that our approach is parametric w.r.t. the actual implementation of both functions `mineWFS` and `minePC`, so that other solutions could be used for implementing them. In particular, a wide range of workflow discovery algorithm exist in the literature which could be exploited to obtain more expressive process models (such as Petri nets, or Event Process Chains). However, such an issue is beyond the scope of this work, yet deserving deeper investigation in the future.

The second part of the algorithm follows a bottom-up merging scheme, somewhat resembling that of agglomerative clustering methods. At each step of the loop (Steps 8-16) a couple of clusters ($\hat{c}_1$ and $\hat{c}_2$) is greedily chosen, such that their respective workflow schemas are the two ones sharing the more behavior, among all those schemas associated with current trace clusters —where behavioral similarity is evaluated here with the approximate measure in Def. 3.

Before merging the selected clusters, however, a check is performed (Step 11), to assess whether such a merge really allows for improving the overall conformance measure fit —evaluated on the `MVPM` gathering all the workflow schemas induced from current trace clusters. Only if the check turns successful, the merge is confirmed, and all the three components of current `MVPM` solution are updated (Steps 12-14). In any case, the couple of clusters chosen is removed from Q (of candidate clusters' pairs), as to avoid considering it in subsequent iterations —this allows to prune the search space of the restructuring phase. It is easy to see that the following property trivially holds, as concerns the correctness of the check in Step 11.

Property. *The conformance of* `MVPM` $\langle CS, \mathcal{W}, \mathcal{R} \rangle$, *measured as* $fit(\langle CS, \mathcal{W}, \mathcal{R} \rangle, L)$, *never decreases during any computation of* `MVPM-mine`. □

It is worth noticing that, in the actual implementation of the algorithm, Q is maintained in the form of an ad-hoc collection of priority queues, one for each cluster in CS, all implemented as heaps. Specifically, for each cluster $c \in CS$, the associated queue stores any other cluster that may still be merged with c, using the respective similarity to c as priority value. Moreover, a dictionary (implemented as a hash table) is also used to support key-based searches over the clusters. In order to ensure fast accesses to the contents of each queue, the extraction of an element is performed by simply marking the corresponding entry as "invalid", without actually removing it from the queue, unless it occupies the top position. Similarly, an update to a priority value is carried out through a "virtual" removal, as explained before, followed by an insertion. Each time a cluster c is extracted from (resp., added to) CS, the corresponding queue is destroyed (resp., created), and c is virtually removed from (resp., added to) the queue of any other cluster. The search for the closest pair of clusters (Step 8) is eventually accomplished by only accessing the top elements of these queues, and then selecting the one with the highest score.

Let n be the number of clusters that were obtained in the first computation phase (Step 3). Since no more than $n-1$ additional clusters can be created in the second phase, any priority queue may contain $O(n)$ (real or invalid) elements. Thus, both the initialization of these queues, and all subsequent accesses to them can be performed in $O(n^2 \times \log n)$.

5.2 System Prototype

Algorithm `MVPM` (cf. Fig. 1) has been fully implemented in a system prototype, in order to apply and validate the approach in practical cases. The logical architecture of the system is depicted in Figure 2.

The *Learning* block is responsible for supporting the discovery of all kinds of models composing a new `MVPM`. In particular, the *Context Data Derivation* module is devised to enrich the vector of each case's data with additional context-related fields, which may allow for better discriminating among the discovered behavioral clusters. Such additional case attributes include both statistics computed on existing attributes – e.g., the `workload` quantifying the the number of total cases currently assigned to each resource or to each high-level organizational entity – as well as new

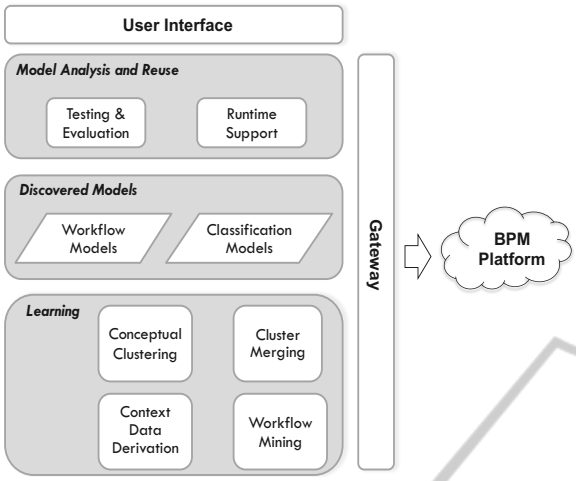

Figure 2: MVPM plug-in architecture.

properties defined by the analyst —e.g., what resource/team was initially allocated to the case.

The traces, enriched with these supplementary attributes, are delivered to the *Conceptual Clustering* module, which groups traces sharing both similar structure and data values, by leveraging some core functionalities of system CLUS (CLUS), a predictive clustering framework supporting the induction of PCT models from tabular data. To this end, a propositional encoding of the traces is built by the module, mixing both context data and structural features (cf. Definition 2).

For each leaf cluster in the PCT, the *Workflow Mining* module extracts a workflow schema, and computes the behavioral profiles associated with it.

Clusters exhibiting similar behaviors (according to the similarity function in Definition 3) are then iteratively merged. The merging procedure, performed by the *Cluster Merging* module, continues until the conformance of the current MVPM is increased, in order to find a good balance between the size of the model, and its ability to effectively describe different behavioral classes. When two clusters are merged, their respective classification rules are merged accordingly, in order to keep the clustering model updated.

For testing purposes and further analysis, both the workflow and the classification models are stored into ad-hoc repositories in the *Discovered Models* block.

As for the *Model Analysis and Reuse* block, the *Testing & Evaluation* model is mainly meant to assess the quality of discovered workflow schemas and associated classification models. In particular, the module supports the computation of different conformance metrics (described in Section 6), based on replaying new test traces through the workflow schema of the cluster they are assigned to.

Finally, the *Runtime Support* module is devoted to provide advanced run-time services, which can turn very useful for supporting the enactment of flexible and dynamic processes. In particular, each discovered classification model can be applied to any partially unfolded (i.e. not finished) process case, in order to assign it to one of the discovered process variants (i.e. trace clusters). The workflow schema associated with the selected cluster can be then presented to the user, as a customized (context-adaptive) process map, describing how the case could proceed (based on how past cases belonging to the same context variant behaved). In addition, this module is capable to dynamically evaluate the degree of compliance between any new (just finished or still ongoing) trace and its reference workflow schema, in order to possibly detect deviating behaviors.

6 EXPERIMENTS

The approach proposed so far has been validated on the log of a real problem management system (named *VINST*), offered by Volvo IT Belgium, as a benchmark dataset for the 2013 BPI Challenge (Steeman, 2013). Precisely, we used 1487 traces spanning from January 2006 to May 2012.

Each log event stores 8 data attributes (namely, `status`, `substatus`, `resource`, `res_country`, `support_team`, `org_line`, and `org_country`, and `functional_division`). Moreover, for each problem case p, two attributes are associated with p's trace: p's `impact` (*medium*, *low*, or *high*), and the `product` affected by p.

In order to enrich each log trace τ with further context data, we extended it with a series of new attributes indicating the support team and the country hosting the solver team appearing in τ's first event (denoted by `firstOrg` and `firstCountry`, respectively), and two other "environmental" variables: a `workload` indicator, storing the total number of problems open in the system when τ started, and several time dimensions (namely, `week-day`, `month` and `year`) derived from the timestamp of τ's first event.

6.1 Evaluation Settings

Several kinds of metrics have been proposed in the literature to evaluate discovered workflow schemas (Buijs et al., 2012). In particular, fitness metrics, quantifying the capability to replay the log, are typically employed as the main evaluation tool, while other kinds of metrics (e.g., precision-oriented ones)

Table 1: Results (avg±stdDev) obtained by MVPM-mine and several competitors. The best value in each column is in bold.

Trace Clustering Method	*Fitness*	*BehPrec*	*#nodes*	*#edges*	*#edgesPerNode*
MVPM-mine (with descriptive structural features)	**0.865±0.010**	0.635±0.009	**6.2±0.0**	**9.3±0.2**	**2.8±0.1**
MVPM-mine (without descriptive structural features)	0.766±0.075	0.612±0.027	8.0±0.0	13.6±0.2	3.4±0.0
ACTITRAC(De Weerdt et al., 2013)	0.317±0.057	**0.667±0.017**	8.2±0.1	14.1±0.4	3.4±0.1
TRMR(Bose and van der Aalst, 2009b)	0.651±0.024	0.604±0.010	7.8±0.1	13.6±0.2	3.5±0.1
A-TRMR(Bose and van der Aalst, 2009b)	0.764±0.027	0.618±0.011	7.7±0.1	13.3±0.2	3.4±0.1
KGRAM(Bose and van der Aalst, 2009a)	0.558±0.026	0.655±0.007	8.0±0.1	13.2±0.2	3.3±0.0

Table 2: Accuracy of the classification models induced from the discovered clusters. Best values per column are in bold.

	With descriptive structural features		*Without descriptive structural features*	
Trace Clustering Method	*Generalization error*	*Cramer's coefficient*	*Generalization error*	*Cramer's coefficient*
MVPM-mine	**0.924**	**0.849**	**0.823**	**0.732**
ACTITRAC(De Weerdt et al., 2013)	0.721	0.135	0.725	0.124
TRMR(Bose and van der Aalst, 2009b)	0.843	0.575	0.693	0.387
A-TRMR(Bose and van der Aalst, 2009b)	0.883	0.534	0.699	0.297
KGRAM(Bose and van der Aalst, 2009a)	0.789	0.592	0.560	0.330

can support finer grain comparisons among schemas with similar fitness scores.

In our tests, the fitness of each discovered heuristics-net schema was specifically measured via the *Improved Continuous Semantics Fitness* defined in (de Medeiros, 2006). Essentially, the fitness score of a schema W w.r.t. a log L (denoted by *Fitness(W,L)*) is the fraction of L's events that W can parse exactly, with a special punishment factor benefitting schemas yielding fewer replay errors in fewer traces.

The behavioral precision of schema W w.r.t. log L, denoted by $BehPrec(W,L)$, is the average fraction of activities that are not enabled when replaying L through W:

$$BehPrec(W,L) = \frac{1}{|\mathcal{A}(W)| \times |traces(L)|} \times \sum_{\tau \in traces(L)} |\{a \in \mathcal{A}(W) | W \text{ did not enable } a \text{ in } \tau\text{'s replay}\}|$$

We preferred these two rough conformance metrics to standard ones defined for Petri-net models, since we experienced long computation times and very low conformance scores every time we applied the latter ones to heuristics nets – this was likely due to the many invisible transitions that tend to be produced when converting them into Petri nets by using the plugins available in ProM.

We also considered three structural-complexity indicators: the numbers of nodes (*#nodes*) and of edges (*#edges*), and the average number of edges per node (*#edgePerNode*).

As competitors, we considered five state-of-the-art approaches: algorithm Actitrac (De Weerdt et al., 2013); the sequence-based and alphabet-based versions of the approach proposed in (Bose and van der Aalst, 2009b) (denoted by TRMR and A-TRMR, respectively), which relies on a mapping of traces onto a space of behavioral patterns (precisely, *tandem repeats* and *maximal repeats*); the approach proposed in (Bose and van der Aalst, 2009a) (denoted by KGRAM), which exploits a k-gram representation of traces, and DWS (Greco et al., 2006), which recursively partitions a log based on a special kinds of (discriminating) sequential patterns capturing unexpected behaviors (w.r.t. current workflow schemas).

Since some of these methods (namely, TRMR, KGRAM and A-TRMR) need to be provided with the desired number of clusters, we instructed them to search for as many trace clusters as those discovered by our approach, while letting the other methods (i.e., Actitrac and DWS) autonomously determine the right number of clusters. When running algorithm DWS, with its default parameter setting, no actual trace partitioning was found for the log (i.e., only one cluster was obtained). Therefore, since our evaluation founds on assessing the capability to recognize (and discriminate among) different behavioral clusters, we will disregard DWS in the remainder of our analysis.

6.2 Test Results

Table 1 shows the quality results obtained by algorithm MVPM-mine, when run according to two alternative settings: *(i)* using, for each log trace, say τ, the whole set of descriptive features defined in Def. 2, namely context properties (i.e. $context(\tau)$ structural features (i.e. $TV(\tau)$), or *(ii)* using only non-structural features (i.e., $context(\tau)$). For each method and each evaluation measure, the average over all trace clusters, performed in 10-fold cross-validation, is shown in the table. In particular, in each clustering test, we

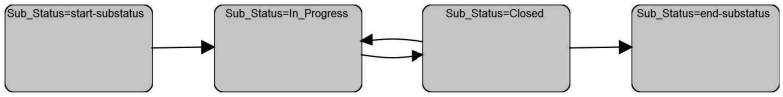

Figure 3: An execution scenario discovered by algorithm MVPM-mine: Workflow Schema of Cluster # 2.

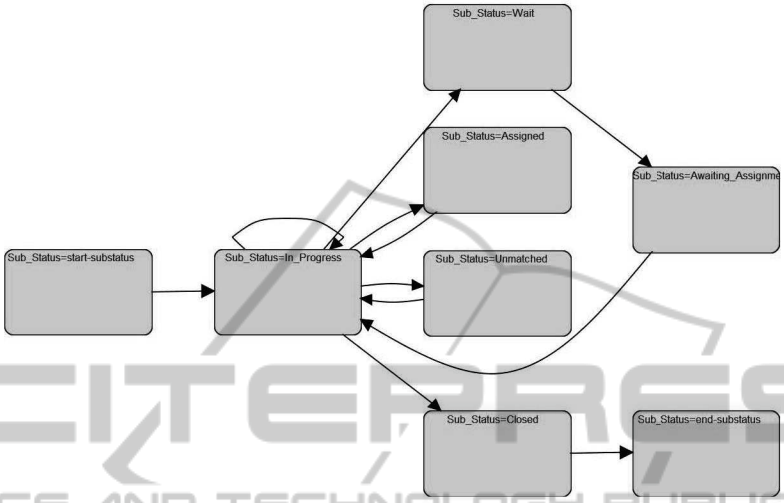

Figure 4: An execution scenario discovered by algorithm MVPM-mine: Workflow Schema of Cluster # 4.

computed an overall *Fitness* (resp., *BehPrec*) measure for each method, as the weighted average of the *Fitness* (resp., *BehPrec*) scores received by the workflow schemas that were induced (with FHM) from all the trace clusters discovered in the test. Since in all cross-validation trials MVPM-mine always found 6 trace clusters, the same number of clusters was given as input to the competitors.

It is easy to see that the version of MVPM-mine exploiting both structural and context features neatly outperforms all competitors in terms of average workflow fitness – the primary comparison metrics – with an outstanding score of 0.865. On the other hand,the achievement of MVPM-mine is quite good also in terms of precision (0.635), if compared to other approaches. The remaining measurements in the table show that such achievements were obtained by way of simple (and readable) workflow schemas: MVPM-mine exhibits, indeed, the lowest average numbers of nodes (6), edges (9) and edges per node (2.8).

Surprisingly good results (comparable to those obtained by its best-performing competitors) were achieved by MVPM-mine even when the descriptive features used for the clustering do not convey any kind of structural information. This result is quite remarkable, as the discovered classification (i.e. conceptual clustering) model can be exploited to predict the structure of any new process case from the very moment when it is started, by only exploiting the knowledge of its associated context data.

This latter impression is confirmed also by the results in Table 2, concerning the capability to accurately discriminate among the discovered clusters. Since none of the competitors directly learn any kind of classification model, for each of these methods, we used the cluster identifier assigned to each trace as it were its associated class label. A logical classification model was the induced with the PCT-learning algorithm implemented in (CLUS) (provided with a single nominal target, instead of a vector of numeric ones, like in function minePC of algorithm MVPM-mine). Again, two different learning settings are considered: *setting a*, where both structural and context data are used as descriptive features; and *setting b*, where only context features are used as descriptive attributes (for classification purposes). For evaluating the accuracy of the discovered classification models, we employed two quality metrics commonly used in classification scenarios: *(i) generalization error* (a.k.a. accuracy), i.e. the number of correct predictions the model performed over the total number of predictions; and *Cramer's coefficient* (Cramer, 1999), gauging the strength of statistical correlation between the real and the predicted classification label.

From Table 2, it is clear that MVPM-mine still outperforms all competitors, so confirming its ability to correctly discriminate different execution scenarios that can be, at a later stage, accurately re-discovered when a classification model is learned over them. It is important to notice that such an ability is kept also

in the critical *setting b*, where no structural information is exploited for separating the discovered clusters. Conversely, all competitors suffers from a neat worsening of performances in this setting.

Table 3: Classification rules describing two distinct trace clusters discovered by MVPM-mine against the whole log.

Trace Cluster	Clustering rule
#2	(product $\in \{PROD278, PROD473, PROD289, \ldots\} \wedge$ firstCountry $\in \{UnitedKingdom, Thailand, \ldots\} \wedge$ firstOrg $\in \{G157\_2nd, T8\_2nd, G186\_2nd, \ldots\} \wedge$ workload $\leq 435) \vee$ (product $\in \{PROD374, PROD729, PROD80, \ldots\} \wedge$ firstOrg $\in \{G349\_3rd, J2\_2nd, G67\_2nd, \ldots\} \wedge$ workload > 477)
#4	substatus=assigned $\wedge$ substatus=wait $\wedge$ substatus=awaiting_assignement $\wedge$ firstOrg $\in \{T17\_2nd, N14\_2nd, N7\_2nd, \ldots\} \wedge$ workload>423

Finally, in order to help the reader get a concrete idea of the kind of knowledge that can be extracted with our approach, Figures 3 and 4 show two workflow models induced from two of the trace trace clusters found in a run of MVPM-mine (launched according to *setting a*). It is easy to notice that the two models are quite different, so allowing to reckon that the handling of problems tended to follow quite different execution scenarios in the analyzed process instances.

For the sake of completeness, we also report, in Table 3, the clustering rules associated with these trace clusters. Notably, these easily-interpretable sets of rules let us identify which specific settings of context variables tended to determine the happening of each of the execution scenarios modeled in Figures 3 and 4.

7 CONCLUSIONS

We have presented a new clustering-oriented process discovery approach, which addresses the critical case of lowly-structured process logs through the induction of a high-quality multi-variant model. The approach is meant to ensure good interpretability of the clusters discovered, and good levels of fitness in the representation of each cluster' behavior. The approach has been implemented into a system prototype, which supports the user in advanced analyses and monitoring tasks.

Tests on a real-life log assess the ability of the approach to find both *(i)* readable workflow models with high levels of behavioral fitness (w.r.t. the respective sublog); and *(ii)* accurate and easy-to-interpret clustering rules, explaining the dependence of process variants on context factors.

As future work, we plan to combine the approach with alternative trace clustering methods (for accomplishing the first of algorithm MVPM-mine), and more sophisticated measures for comparing workflow schemas and for estimating their conformance. We will also intend to test our approach on other real (lowly-structured) business process logs, possibly featuring a richer range of non-structural data.

REFERENCES

CLUS: A predictive clustering system. *http://dtai.cs.kuleuven.be/clus/*.

Alves de Medeiros, A. K., van der Aalst, W. M. P., and Weijters, A. J. M. M. (2008). Quantifying process equivalence based on observed behavior. *Data & Knowledge Engineering*, 64(1):55–74.

Blockeel, H. and Raedt, L. D. (1998). Top-down induction of first-order logical decision trees. *Artificial Intelligence*, 101(1-2):285–297.

Bose, R. P. J. C. and van der Aalst, W. M. P. (2009a). Context aware trace clustering: Towards improving process mining results. In *Proc. of SIAM International Conference on Data Mining*, pages 401–412.

Bose, R. P. J. C. and van der Aalst, W. M. P. (2009b). Trace clustering based on conserved patterns: Towards achieving better process models. In *Business Process Management Workshops*, pages 170–181.

Buijs, J., van Dongen, B., and van der Aalst, W. (2012). On the role of fitness, precision, generalization and simplicity in process discovery. In *On the Move to Meaningful Internet Systems: OTM 2012*, volume 7565, pages 305–322.

Cramer, H. (1999). *Mathematical Methods of Statistics*. Princeton University Press.

de Medeiros, A. A. (2006). *Genetic Process Mining*. Phd thesis, Eindhoven University of Technology.

De Weerdt, J., van den Broucke, S., Sand Vanthienen, J., and Baesens, B. (2013). Active trace clustering for improved process discovery. *IEEE Trans. on Knowl. and Data Eng.*, 25(12):2708–2720.

Ferreira, D., Zacarias, M., Malheiros, M., and Ferreira, P. (2007). Approaching process mining with sequence clustering: Experiments and findings. In *Proc. of 5th Int. Conf. on Business Process Management (BPM'07)*, pages 360–374.

Folino, F., Greco, G., Guzzo, A., and Pontieri, L. (2008). Discovering multi-perspective process models: The case of loosely-structured processes. In *ICEIS 2008, Revised Selected Papers*, pages 130–143.

Folino, F., Greco, G., Guzzo, A., and Pontieri, L. (2011). Mining usage scenarios in business processes: Outlier-aware discovery and run-time prediction. *Data & Knowledge Engineering*, 70(12):1005–1029.

Folino, F., Guarascio, M., and Pontieri, L. (2012). Discovering context-aware models for predicting business process performances. In *Proc. of 20th Intl. Conf. on Cooperative Information Systems (CoopIS'12)*, pages 287–304.

Greco, G., Guzzo, A., Pontieri, L., and Saccà, D. (2006). Discovering expressive process models by clustering log traces. *IEEE Trans. Knowl. and Data Eng.*, 18(8).

Kunze, M., Weidlich, M., and Weske, M. (2011). Behavioral similarity: A proper metric. In *Proc. of 9th Int. Conf. on Business Process Management (BPM'11)*, pages 166–181.

Rozinat, A. and van der Aalst, W. M. P. (2008). Conformance checking of processes based on monitoring real behavior. *Information Systems*, 33(1):64–95.

Song, M., Günther, C. W., and van der Aalst, W. (2008). Trace clustering in process mining. In *Proc. of Business Process Management Workshops (BPI'08)*, pages 109–120.

Steeman, W. (2013). BPI challenge 2013, closed problems.

van der Aalst, W. (1998). The application of Petri nets to worflow management. *Journal of Circuits, Systems, and Computers*, 8(1):21–66.

van Der Aalst, W., Adriansyah, A., and van Dongen, B. (2011). Causal nets: a modeling language tailored towards process discovery. In *Proc. of the 22nd Intl. Conf. on Concurrency Theory*, CONCUR'11, pages 28–42, Berlin, Heidelberg. Springer-Verlag.

van der Aalst, W. M. P., van Dongen, B. F., Herbst, J., Maruster, L., Schimm, G., and Weijters, A. J. M. M. (2003). Workflow mining: a survey of issues and approaches. *Data & Knowledge Engineering*, 47(2):237–267.

Weidlich, M., Polyvyanyy, A., Desai, N., Mendling, J., and Weske, M. (2011). Process compliance analysis based on behavioural profiles. *Information Systems*, 36(7):1009–1025.

Weijters, A. J. M. M. and Ribeiro, J. T. S. (2011). Flexible heuristics miner (FHM). In *Proc. of IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2011)*, pages 310–317.

Weijters, A. J. M. M. and van der Aalst, W. M. P. (2003). Rediscovering workflow models from event-based data using Little Thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162.