Business Process Variant Analysis: Survey and Classification

FARBOD TAYMOURI, The University of Melbourne, AUSTRALIA
MARCELLO LA ROSA, The University of Melbourne, AUSTRALIA
MARLON DUMAS, University of Tartu, ESTONIA
FABRIZIO MARIA MAGGI, University of Tartu, ESTONIA

It is common for business processes to exhibit a high degree of internal heterogeneity, in the sense that the executions of the process differ widely from each other due to contextual factors, human factors, or deliberate business decisions. For example, a quote-to-cash process in a multinational company is typically executed differently across different countries or even across different regions in the same country. Similarly, an insurance claims handling process might be executed differently across different claims handling centres or across multiple teams within the same claims handling centre. A subset of executions of a business process that can be distinguished from others based on a given predicate (e.g. the executions of a process in a given country) is called a process variant. Understanding differences between process variants helps analysts and managers to make informed decisions as to how to standardize or otherwise improve a business process, for example by helping them find out what makes it that a given variant exhibits a higher performance than another one. Process variant analysis is a family of techniques to analyze event logs produced during the execution of a process, in order to identify and explain the differences between two or more process variants. A wide range of methods for process variant analysis have been proposed in the past decade. However, due to the interdisciplinary nature of this field, the proposed methods and the types of differences they can identify vary widely, and there is a lack of a unifying view of the field. To close this gap, this article presents a systematic literature review of methods for process variant analysis. The identified studies are classified according to their inputs, outputs, analysis purpose, underpinning algorithms, and extra-functional characteristics. The paper closes with a broad classification of approaches into three categories based on the paradigm they employ to compare multiple process variants.

CCS Concepts: • Applied computing → Process mining.

Additional Key Words and Phrases: business process management, process mining, machine learning

ACM Reference Format:
FARBOD TAYMOURI, MARCELLO LA ROSA, MARLON DUMAS, and FABRIZIO MARIA MAGGI. 2019. Business Process Variant Analysis: Survey and Classification. 1, 1 (January 2019), 29 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Process mining [62] is a body of methods and tools to analyze business process execution logs (called event logs), in order to extract insights about possible performance deficiencies and improvement opportunities. In this context, an event log is a collection of traces, each one consisting of the sequence of events recorded during the execution of one process instance (herein called a case).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM.

XXX-XXXX/2019/1-ART $15.00
https://doi.org/10.1145/1122445.1122456
Depending on their inputs and their outputs, the following categories of process mining techniques can be distinguished [23]:

- Automated process discovery techniques, which allow one to discover a business process model from an event log.
- Conformance checking techniques, which allow one to compare a process model against an event log in order to qualify and quantify their differences.
- Performance mining techniques, which allow one to enhance a given process model with performance information extracted from an event log.
- Variant analysis techniques, which allow one to compare two or more event logs corresponding to different variants of a business process, in order to qualify their differences.

This article deals with the latter category of techniques. The goal of business process variant analysis is to help business analysts to understand why and how multiple variants of a process differ. In this setting, a process variant is a subset of executions of a business process that can be distinguished from others based on some characteristic. For example, if a process is executed in three countries, say C1, C2 and C3, we can distinguish three variants of this process: one for each of these countries.

Given an event log of a business process, a process variant takes the form of a set of traces (herein called a cohort) that can be separated from others based on a predicate, i.e. a function that maps each trace in the log to a boolean variable. The first step in process variant analysis is to split the event log into cohorts using a trace filtering operation. In the above scenario, the predicate that characterizes the first variant is ‘country = C1’. By applying a log filter that retains only those traces for which this predicate holds, we can extract the cohort corresponding to the first process variant, and similarly for the other two variants.

Given that an event log has been split into multiple cohorts, relevant questions that variant analysis seeks to answer include: why do the executions of a given cohort take longer to complete, on average, than those of another cohort? Or what activities are often skipped in one cohort but are never or seldom skipped in another cohort?

As hinted by these questions, variant analysis techniques may cover different perspectives of a business process, including the following ones:

- **Control flow**: Along this perspective, the variants are compared in terms of the occurrence of activities in the execution traces and their relative execution order.
- **Performance**: Along this perspective, the variants are compared in terms of performance characteristics or performance measures.

The above considerations are depicted in Figure 1, which shows that variant analysis starts by splitting an event log into multiple cohorts, which are then compared according to different perspectives, including the control-flow and the performance perspectives.

A wide range of methods for log-based process variant analysis have been proposed in the past decade. However, due to the interdisciplinary nature of this field, the proposed methods and the types of differences they can identify vary widely, and there is a lack of a unifying view of the field. To close this gap, this article presents a systematic literature review of methods for process variant analysis. The article also proposes a taxonomy of existing methods and identifies gaps in the field.

The article is organized as follows. Section 2 introduces background concepts and terminology used in subsequent sections. Following that, Section 3 describes the search and selection criteria for identifying relevant studies. Next, Section 4 provides an in-depth analysis and detailed classification of the identified studies. Section 5 presents a broader classification of approaches in terms of the paradigms employed to compare process variants. Finally, Section 6 summarizes the findings.
2 PRELIMINARIES AND BACKGROUND

Process variant analysis, as we will explain in the upcoming sections, has been tackled in two different fields: process mining and machine learning. This section provides basic concepts that will help us to explain how process variant analysis has been approached in each of these the two fields.

2.1 Process mining

Process mining is a research area between Business Process Management (BPM) and data science that is concerned with deriving useful insights from process execution data. Process mining techniques can support various phases of the BPM life-cycle, such as process discovery, process analysis and process monitoring [62]. In fact, it aims at discovering, monitoring and improving real processes by extracting knowledge from event logs readily available in today’s information systems [62]. The recent significant growth of event data available on the one side and the development of mature process mining techniques on the other side are pushing companies and organizations to exploit process mining to analyze and improve their processes.

The input artifacts for process mining are a process model and an event log. A process model shows the expected behaviour of the process, and the event log shows the process executions, a.k.a. footprint or observed behavior. Process mining techniques can be classified into three types. The first type, discovery, aims at discovering a process model from an event log without using any a-prior information. The second type, conformance checking, focuses on confronting an event log and a process model (discovered from an event log or manually designed). Conformance checking is used to check if reality, as recorded in the log, conforms to the model and vice versa. The third type, Enhancement, intends to improve an existing process model by using the information about the actual process executions recorded in the event log, or the disconformities identified via conformance checking.

An event log consists of cases or traces, each capturing a particular execution of a business process. Each case consists of a number of events and each event represents the execution of a particular activity in the process. Each event has a range of attributes of which three are mandatory: i) the case identifier specifying which case generated this event, ii) the event class (or activity name) indicating which activity the event refers to, and iii) the timestamp indicating the completion time of the activity. Note that, in process mining approaches, the completion time of each event
determines the order of the events. We call performance attributes all the other attributes different from the ones mentioned above.

| Case attributes | Event attributes |
|-----------------|------------------|
| Id   | City | Sex | Product | Activity | Completion time | Starting time | Resource |
| 1    | NY   | M   | Book    | Order    | 1/1/2017 9:13:00 | 1/1/2017 9:12:57 | David    |
| 1    | NY   | M   | Book    | Pay in cash | 1/1/2017 9:14:20 | 1/1/2017 9:14:10 | John     |
| 1    | NY   | M   | Book    | Approval  | 1/1/2017 9:16:00 | 1/1/2017 9:15:37 | Tiffany  |
| 2    | MA   | F   | Sofa    | Order    | 2/1/2017 16:55:00 | 2/1/2017 16:45:00 | Joe      |
| 2    | MA   | F   | Sofa    | Pay by card | 2/1/2017 17:00:00 | 2/1/2017 16:58:00 | Nathan   |
| 2    | MA   | F   | Sofa    | Disapproval | 3/1/2017 9:00:00 | 3/1/2017 8:57:00 | Jane     |
| 2    | MA   | F   | Sofa    | Pay in cash | 3/1/2017 9:01:50 | 3/1/2017 9:01:20 | John     |
| 2    | MA   | F   | Sofa    | Approval  | 3/1/2017 9:03:10 | 3/1/2017 9:02:12 | Luis     |
| 3    | LA   | M   | T.V.    | Order    | 1/1/2017 9:13:00 | 1/1/2017 9:10:00 | James    |
| 3    | LA   | M   | T.V.    | Pay by card | 1/1/2017 9:24:20 | 1/1/2017 9:22:20 | Patrick  |
| 3    | LA   | M   | T.V.    | Approval  | 1/1/2017 9:26:00 | 1/1/2017 9:14:00 | Carla    |
| 4    | LA   | F   | Book    | Order    | 1/1/2017 7:13:00 | 1/1/2017 7:10:00 | James    |
| 4    | LA   | F   | Book    | Pay in cash | 1/1/2017 7:24:20 | 1/1/2017 7:22:20 | Patrick  |
| 4    | LA   | F   | Book    | Approval  | 1/1/2017 7:26:00 | 1/1/2017 7:25:00 | Carla    |

Table 1. A sample of an event log from an online retailer

For example, Table 1 shows an event log for a simplified online shopping process from a retailer. A case in this table has four (case) attributes, Id, City (the place where the buyer lives), Sex (of the buyer), and Product. Also, each event has several (event) attributes such as Activity, Starting time, Completion time, and Resource (who processes the activity from the retailer side). The order of activities inside a case is called control flow. For instance, in the first case (Id=1), the customer starts by ordering a book (Order), then he pays in cash (Pay in cash), and finally the retailer approves the payment (Approval).

We now define the mentioned concepts formally.

**Definition 2.1 (Event).** An event is a tuple $(a, c, t, (d_1, v_1), \ldots, (d_m, v_m))$ where $a$ is the activity name, $c$ is the case id, $t$ is the timestamp and $(d_1, v_1), \ldots, (d_m, v_m)$ (where $m \geq 0$) are the event or case attributes and their values.

The universe of all events is hereby denoted by $E$. If we consider Table 1, an event is $(\text{Order}, 1, 1/1/2017 9:13:00, (\text{Starting time}, 1/1/2017 9:12:57), (\text{Resource}, \text{David}))$.

The sequence of events generated by a given process execution forms a trace. Formally:

**Definition 2.2 (Trace).** A trace is a non-empty sequence $\sigma = [e_1, \ldots, e_n]$ of events such that $\forall i \in [1..n], e_i \in E$, and $\forall i, j \in [1..n] e_i.c = e_j.c$. In other words, all events in the trace refer to the same case id.

The corresponding trace for the first case is $[(\text{Order}, 1, 1/1/2017 9:13:00, (\text{Starting time}, 1/1/2017 9:12:57), (\text{Resource}, \text{David})), (\text{Pay in cash}, 1, 1/1/2017 9:14:20, (\text{Starting time}, 1/1/2017 9:14:10), (\text{Resource}, \text{John})), (\text{Approval}, 1, 1/1/2017 9:16:00, (\text{Starting time}, 1/1/2017 9:15:37), (\text{Resource}, \text{Tiffany}))]$.

A set of traces is called an event log. Also, we can create process variants based on case attributes such as Sex, Product, or the cycle time of a case.
Definition 2.3 (Process cohort (or Process variant)). An event log \( L \) can be partitioned into a finite set of groups called process variants or process variants \( \xi_1, \xi_2, \ldots, \xi_n \), such that \( L = \xi_1 \cup \xi_2 \cup \ldots \cup \xi_n \), and \( \forall i, j, \xi_i \cap \xi_j = \emptyset \), and, \( \exists d \) such that \( \forall \xi_k \) and \( \forall \sigma_i, \sigma_j \in \xi_k, \sigma_i.d = \sigma_j.d. \)

The above definition of a process variant emphasizes that the process executions in the same group must share the same attribute value for a given attribute, and each process execution belongs only to one process variant.

A process model is a graphical entity used to represent how a process is executed in an organization. In the business domain, a business process model is a collection of inter-related events, activities, and decision points that involve a number of actors and objects, which collectively lead to an outcome that is of value for a customer [23]. Companies and organizations usually use different notations to represent their business process models and each of them has different characteristics. Thus, selecting an appropriate process modeling language is essential. However, it is worth mentioning that often one formalism can easily be translated to other notations [63]. In the following, we present a short introduction to Petri nets [43] and transition systems [62], the most used notations to formally represent business process models.

A Petri net \( N = (P, T, F) \) is a directed graph with a set \( P \) of nodes called places and a set \( T \) of transitions. Places are represented by circles and transitions by squares. The nodes are connected via directed arcs \( F \subseteq (P \times T) \cup (T \times P) \). Connections between two nodes of the same type are not allowed. Given a transition \( t \in T \), \( t' \) is used to indicate the set of input places of \( t \), which are the places \( p \) with a directed arc from \( p \) to \( t \) (i.e., such that \( (p, t) \in F \)). Similarly, \( t^* \) indicates the set of output places, namely the places \( p \) with a direct arc from \( t \) to \( p \). At any time, a place can contain zero or more tokens, drawn as black dots. The state of a Petri net, a.k.a. marking \( m \), is determined by the number of tokens in places, i.e., \( m : P \rightarrow N \).

In any run of a Petri net, the number of tokens in places (i.e., the marking) may change. A transition \( t \) is enabled at a marking \( m \) iff each input place contains at least one token, i.e., \( \forall p \in {}^*t, M(p) > 0 \). A transition \( t \) can fire at a marking \( m \) iff it is enabled. As result of firing a transition \( t \), one token is “consumed” from each input place and one is “produced” in each output place. This is denoted as \( m \xrightarrow{t} m' \).

![Fig. 2. Labeled Petri net, transitions are squares, places are circles and tokens are black dots](image)

For example, consider the process model in Figure 2 reflecting the behavior of the event log in Table 1. The set of transitions and places are \( \{t_1, t_2, t_3, t_4, t_5\} \) and \( \{p_1, p_2, p_3, p_4\} \), respectively. Also, the labeling function is \( \ell(t_1) = \text{"Order"}, \ell(t_2) = \text{"Pay in cash"}, \ell(t_3) = \text{"Pay by card"}, \ell(t_4) = \text{"Approval"}, \ell(t_5) = \text{"Disapproval"}. \) In the process model, only \( p_1 \) has one token, i.e., \( m[p_1] = 1 \), moreover, \( t_1 \) is enabled and ready to fire. To show how the model executes, suppose that \( t_1 \) fires, then it consumes one token from \( p_1 \) and produces one token into \( p_2 \), thus, \( t_2 \) and \( t_3 \) become enabled; however, only one of them can fire. After firing \( t_2 \) or \( t_3 \), then one token is placed in \( p_5 \), which
enables $t_4$ and $t_5$. Finally, one of $t_5$ or $t_4$ is fired, where the former marks $p_2$ and the execution continues, whereas the latter marks $p_4$ and the execution terminates.

Fig. 3. A transition system having one initial state and one final state

A transition system is a triplet $TS = (S, A, T)$, where $S$ is the set of states, $A \subseteq \Sigma$ is the set of activities (often referred to actions), and $T \subseteq S \times A \times S$ is the set of transitions. $S^{start} \subseteq S$ is the set of initial states, and $S^{end} \subseteq S$ is the set of final states. A transition system is the most basic process modeling formalism compared to other notations; it is also known as a Directed Graph (DG). As an example, consider the transition system in Figure 3 reflecting the behavior of the event log in Table 1. The corresponding set of states and activities are $S = \{s_1, s_2, s_3, s_4, s_5\}$, and $A = \{\text{Order, Pay in cash, Pay by card, Disapproval, Approval}\}$. Also, $S^{start} = s_1$, and $S^{end} = s_5$.

Two important concepts that would be helpful in variant analysis of process executions are the notions of replaying [62] and alignment [1]. Replaying a process execution on a process model means to rerun the process execution on the process model to quantify discrepancies between them. Though replaying provides useful and easy-to-understand information, a more fundamental way to identify such deviations is by using alignments. Alignments play an important role in conformance checking. Given a process model and a process execution, an alignment quantifies to what extent the process model can mimic the process execution. An alignment is a two-row matrix that lines up corresponding activities in the process model and in the process execution. Formally:

Definition 2.4 (Alignment). Given a process model and a process execution, let $\Sigma$ be the universe of all activities. Let $A_M \subseteq \Sigma$ and $A_L \subseteq \Sigma$ be the alphabet of activities in the model and events in the event log, respectively, and $\bot$ the empty set, then an alignment, denoted by $\alpha$, is a sequence of legal moves, where:

- $(x, y)$ is a synchronous move if $x \in A_L$, $y \in A_M$
- $(x, y)$ is a move in log if $x \in A_L$ and $y = \bot$.
- $(x, y)$ is a move in model if $x = \bot$ and $y \in A_M$.
- $(x, y)$ is an illegal move, otherwise.

For example, an alignment between the process execution $\sigma_L = [\text{Order, Approval, Pay by card}]$, and the process model in Figure 2, with initial marking and final marking denoted with $m_i$ (a single token in $p_1$) and $m_f$ (a single token in $p_4$), is the following:

| $\alpha$ | $\text{Order}$ | $\text{Approval}$ | Pay by card | $\bot$ | Pay by card | Approval |
| --- | --- | --- | --- | --- | --- | --- |
| Order | $\bot$ | $\text{Pay by card}$ | Approval |

In this example (Order, Order) and (Pay by card, Pay by card) are synchronous moves, and (Approval, $\bot$) and ($\bot$, Approval) are move in log and model respectively, or, in short, asynchronous moves. Note that, ignoring all occurrences of $\bot$, the projection on the first element of the moves yields $\sigma_L$ and the projection on the second one yields a sequence $\sigma''$ such that $m_i \xrightarrow{\sigma''} m_f$. Generally
speaking, a move in log for a transition $t$ indicates that $t$ occurred when not allowed; a move in model for a transition $t$ indicates that $t$ did not occur, when, conversely, expected. An alignment usually is quantified with a fitness value, which, in the simplest case, is the number of synchronous moves divided by the total number of moves. For the mentioned example, the fitness is $\frac{2}{4}$.

### 2.2 Machine learning

Machine learning is the systematic design, analysis and study of algorithms and systems that learn from past experiences. Machine learning is inherently a multidisciplinary field. It draws on results from artificial intelligence, probability and statistics, computational complexity theory, control theory, information theory, philosophy, psychology, neurobiology, and other fields [42].

Given a problem at hand, the first step in learning from data is to have related observations. The raw observations comprise multidimensional data, event log data, graph data, and other types of data. Moreover, for every type of data, several sophisticated machine learning algorithms have been proposed by researchers. However, because of historical and technical reasons, most of the developed algorithms use multidimensional data or encode other types of data into a multidimensional representation. In an $n$-dimensional representation, every entity is shown as a vector of length $n$, and each dimension is called a feature or attribute. Thus, a group of observations $D$ can be shown as a multiset of vectors as follows:

$$D = \{x_i\}_{i=1}^m$$ or $$\{x_1, x_1, \ldots, x_m\}, \text{ where } \forall i \in [1..m], \ x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,n})$$

In the above representation, $x_i$ is a vector with $n$ features $x_{i,1}, x_{i,2}, \ldots, x_{i,n}$. A feature can be a complex structured object, such as an image, a sentence, a time series, a molecular shape, a graph, a sequence prefix [44].

Broadly speaking, a machine learning task can be of two types:

- In descriptive or unsupervised learning approaches, given a set of observations $D$, the objective is to find interesting patterns in the data. A canonical example of unsupervised learning is the problem of clustering data observations into groups.
- In supervised learning or predictive approaches, each vector $x_i$ has an associate label $y_i$, which is called response variable. Response variables can be of different nature, but the most methods assumes that it is categorical or real-valued. The set of labeled vectors, i.e., $D = \{(x_i, y_i)\}_{i=1}^m$ is called the training set, and the main objective of supervised learning algorithms is to estimate a mapping function from $x$ to $y$, i.e., $y = f(x)$. The estimated function or the trained model is called a classification model for categorical response variables, and a regression model for real-valued response variables.

There exist many well-developed and dedicated algorithms for the machine learning approaches just mentioned. For example, decision tree and rule-based algorithms and their variants are among the first proposed supervised learning algorithms. A decision tree, using a set of hierarchical decisions on the features, constructs a tree-like structure to classify an input observation. Similarly, a rule-based classifier uses a set of “if-then” rules to match antecedents to consequents. A rule is expressed as follow:

$$\text{IF Antecedent THEN Consequent}$$

where the antecedent is a logical combination of features, e.g., $(x_{i,1} \land x_{i,2}) \lor x_{i,4}$, and the consequent is the class label. Rule-based algorithms are the supervised version of association rule mining algorithms, which determine relationships in a set of observations.

Though decision tree and rule-based classifiers adopt different underlying mechanisms for classification tasks, a decision tree may be viewed as a particular case of a rule-based classifier in which each path of the decision tree corresponds to a rule.
From the probabilistic perspective, despite the variety of proposed supervised and unsupervised learning algorithms, either try to approximate probability values. In particular, supervised learning algorithms strive to approximate $p(y_i|x_i)$, i.e., the probability of a class label given an input vector, whereas an unsupervised algorithm can be viewed as a density estimation, i.e., $p(x_i)$ [44]. The differences among machine learning algorithms are in the way they compute these probabilities.

The performance of a machine learning algorithm can be evaluated in different ways. For unsupervised learning algorithms, the validation is often difficult since the problem is defined in a descriptive way. However, some validation criteria can be defined to evaluate the objective function upon which observations are clustered together. In contrast, the predictive ability of a supervised learning algorithm can be evaluated using the input labels. For example, accuracy and Area Under Curve (AUC) can be used to evaluate a classification model. The former shows the ratio of the number of correct predictions to the total number of predictions, and the latter, for a binary classification model, provides the probability that a model ranks a random positive example more highly than a random negative example.

A learning paradigm that has received much attention over the past few years is the learning by committee or ensemble learning [22]. Ensemble learning is motivated by the fact that, given a problem, different learning algorithms might provide different results due to the specific characteristics of the underlying learning algorithms, or their sensitivity to the random artifacts in the input. Therefore, the goal of ensemble learning is to combine the results from multiple learners to improve the quality of the results. In unsupervised learning approaches, it is evident that there are many alternative solutions, i.e., clustering models, alongside a large number of validation criteria, and no single model or validation criterion provides the optimal clustering. Thus, ensemble clustering, proposed by [55], combines many clustering models to create a more robust clustering approach. By the same token, in supervised learning, a set of base learners is created and trained in different ways, and then the results of base learners are combined to create the final prediction. A very simple way to combine outputs of base learners, for real-valued outputs, is to average them:

$$f(x) = \frac{1}{p} \sum_{j=1}^{p} f_j(x)$$

(3)

In the above expression, there are $p$ base learners, and $f_j(x)$ is the output of the $j$-th base learner.

![Fig. 4. Trade-off between accuracy and explainability of learning algorithms](https://www.darpa.mil/attachments/XAIProgramUpdate.pdf?source=post_page)
Notwithstanding the importance of the accuracy of a machine learning algorithm, an algorithm can also be evaluated from other perspectives. For example, in several situations, it is necessary to have an explainable machine learning model. Explainability is defined as the science of understanding what a model did, or might have done [29]. More simply, explainability is the extent to which the internal mechanics of a machine learning system can be explained in human terms. The concept of explainability can be applied to all supervised and unsupervised learning approaches. For example, decision tree and rule-based classifiers are highly explainable, i.e., the internal structure of a decision tree and a set of rules can be easily explained in human terms; on the other hand, the internal structure of ensemble models is very difficult to grasp in human terms. Although both accuracy and explainability are two important aspects of a machine learning algorithm, they interfere with each other. Indeed, the internal structure of a sophisticated machine learning algorithm that comes up with very high accuracy is hardly explainable in human terms and it acts as a black-box. In this sense, according to the no free lunch theorem, there is no universal best model [67]. Figure 4 presents the trade-off between accuracy and explainability aspects for well-known machine learning algorithms.

3 SEARCH METHODOLOGY

We conducted a Systematic Literature Review (SLR) of process variant analysis methods, by following the SLR guidelines in [36]. In line with these guidelines, we started by posing a research question to clarify the goals of the search. From the research question, a search string was derived for retrieving related documents from academic digital libraries. The following subsections detail the SLR steps followed in this paper.

3.1 Research Question
The main aim of this paper is to review proposed methods for process variant analysis. Process variant analysis is a rather broad topic. Therefore, to confine our search space, we defined the following research question (RQ): Given a set of event logs of two or more variants of process, how to identify and explain the differences among these variants?

3.2 Study Retrieval and Selection
To retrieve relevant papers based on RQ, the following keywords were considered:

- “event log” - a relevant study must consider event logs as inputs;
- “process variant analysis” - a relevant study should concern the analysis of the executions of a process;
- “process variants comparison” - a relevant study should concern the comparison of sets of process executions;

Though the aforementioned terms are the most related keywords, we realized that some works related to process variant analysis use the term “deviance mining” to indicate this type of analysis; therefore, we included additional terms, namely, “process deviance mining” and “process deviance comparison”, to cover such works.

Using these keywords we derived a search string that was submitted to Google Scholar. Google Scholar is the world’s largest academic search engine, which encompasses other academic databases like ACM Digital Library and IEEE Xplore [32]. The retrieved documents are those that have at least one of the above terms in their title, keywords or the main body of the paper.

The search resulted in 88 unique articles published between January 2000 and April 2019. Figure 5 shows the number of publications per year according to the proposed search query. One can see
an upward trend for the research publications on process variant analysis. This shows that this area of research is recently getting more and more attention.

![Number of publications per year](image)

**Fig. 5. Number of publications per year**

To eliminate irrelevant results and to avoid exploring marginal studies without any follow-up, we applied the following inclusion criteria:

- **INC**₁: The study is about variant analysis of processes (this criterion was assessed by reading title and abstract).
- **INC**₂: The study is cited at least five times (this threshold was relaxed for publications from 2018 onward where instead of considering the number of citations we considered, as criterion, the number of pages, i.e., to have at least ten pages single-column or five pages double-column).

We intentionally kept **INC**₁ open by using only the term “process”. In this way, we can cover different types of processes such as business processes and software development processes.

After applying the above inclusion criteria, we obtained 14 relevant studies. To increase the sensitivity of our research, we proceeded with the Snowball sampling method [7], i.e., we retrieved the papers that are related to (cite or are cited by) these 14 studies and re-applied the same inclusion criteria as above. This procedure resulted in 363 papers, of which we retained 91 unique papers after re-applying the inclusion criteria.

The list of studies that passed the inclusion criteria were further assessed according to a number of exclusion criteria:

- **EX**₁: The study does not propose a concrete technique for comparison of process variants.
- **EX**₂: The proposed technique focuses on building predictive models that can generate predictions based on running process instances, as opposed to supporting the (post-mortem) comparison of process variants.
- **EX**₃: The technique does not take an event log as input.

More precisely, **EX**₁ excludes those works that are not related to proposing a method for analyzing or comparing process variants. The second exclusion criteria **EX**₂ eliminates works that are focused
on predictive process monitoring techniques. The main focus of these latter studies is on predicting future states of ongoing cases, rather than comparing characteristics of sets of completed cases. In addition, predictive monitoring techniques have been studied extensively in previous surveys [21, 45, 60, 64]. The last exclusion criteria EX3 leaves out those studies that do not use event logs as input. These might be studies that compare process models represented using different formalism. Though these approaches might be inspiring for process variant analysis, the scope of this paper is limited to review the current existing techniques that leverage process executions. The application of the exclusion criteria resulted in 29 relevant studies out of 91 works selected in the previous step.²

4 ANALYSIS AND CLASSIFICATION OF METHODS

Research question RQ can be answered by categorizing the selected works using different dimensions specifying the typology of the existing methods and their characteristics. In particular, each study can be decomposed into the following dimensions:

- Input data
- Outcome
- Process perspective (control flow, resources, data)
- Family of algorithms (the main algorithm used in the study)
- Evaluation data (real-life or artificial logs) and application domain (e.g., insurance, banking, healthcare)
- Implementation (standalone or plug-in, and tool accessibility)

Table 2 provides an overview of the identified studies according to the mentioned dimensions. In the following, we provide an overview of each study and, then, more details about the classification for each dimension.

4.1 Overview

According to our results, the work in [51] is the first work that considers process variant analysis at the process execution level. A process execution, in this work, contains treatment activities that a hospital applies to breast cancer patients. This work aims at gaining a deeper understanding of an existing breast cancer care process to discover process inefficiencies, exceptions and variations, and to find their root causes. To this end, Hidden Markov Models are used for process discovery and Formal Concept Analysis [28] is employed to analyze clusters of patients identified in the discovered processes.

Similarly to this work, a series of interactive tools for extracting and visualizing clinical care pathways is presented in [40]. The work considers a process execution as a sequence of clinical activities that patients receive in their care journeys. The main objective of the paper is to examine the impact and correlation of clinical activities on the clinical care pathway of a patient for specific diseases. Different techniques like frequent pattern mining and trace clustering are applied to accomplish this goal. In this study, a tool for visualizing the results of the analysis is also presented. The tool discovers dependency graph models using the Heuristic Miner [66], and then the impactful patterns obtained from frequent pattern mining are superimposed to them to highlight differences among different variants.

²All the retrieved papers can be found at https://figshare.com/articles/Selected_articles/9999887
| Study                  | Year | Input data                  | Outcome                          | Algorithm                                      | Domain             | Implement. | Type of analysis                  |
|-----------------------|------|-----------------------------|----------------------------------|-----------------------------------------------|--------------------|------------|-----------------------------------|
| Poelmans et al. [51]  | 2010 | Event log                   | Rule, Descriptive statistics     | Hidden Markov Model, Formal Concept Analysis  | Healthcare         | Matlab     | Control flow, Cycle time          |
| Swinnen et al. [59]   | 2012 | Event log, Process model    | Rule                             | Fuzzy mining, Association rule mining         | Financial          | Weka, ProM | Control flow                      |
| Buijs et al. [14]     | 2012 | Event log                   | Alignment matrix, Descriptive statistics | Alignment analysis                           | Public administration | ProM | Control flow                      |
| Suriadi et al. [58]   | 2013 | Event log                   | Rule, Process model (TS), Descriptive statistics | Fuzzy mining, Causal relation analysis     | Insurance          | Weka, ProM, Disco | Control flow                      |
| Sun et al. [56]       | 2013 | Event log                   | Rule                             | Contrast itemset mining                       | Financial, Industrial | MOSPER | Control flow                      |
| Bose et al. [12]      | 2013 | Event log                   | Rule                             | Decision tree induction, Association rule mining | Industrial | ProM | Control flow                      |
| Lakshmanan et al. [40]| 2013 | Event log                   | Rule, Process model (TS)         | Frequent pattern mining, Clustering           | Healthcare          | RPI        | Control flow                      |
| Krigstein et al. [37] | 2013 | Event log, Process model    | Annotated process model (DG)     | Difference Model analysis                     | Logistic            | N/A        | Control flow                      |
| Suriadi et al. [37]   | 2014 | Event log                   | Process model (PN), Alignment matrix, Descriptive statistics | Fuzzy mining, Clustering                       | Healthcare          | RapidMiner, Weka, ProM | Control flow, Waiting time between activities |
| Buijs et al. [13]     | 2014 | Event log, Process model    | Alignment matrix, Descriptive statistics | Alignment analysis                           | Public administration | ProM | Control flow                      |
| Partington et al. [49]| 2015 | Event log                   | Annotated process model (BPMN, TS) | Fuzzy mining, Log replay                       | Healthcare          | ProM, Nitro | Control flow, Waiting time between activities, Cycle time |
| Cordes et al. [16]    | 2015 | Event log                   | Annotated process model (DG)     | Difference Model analysis (using TGraph)      | Healthcare          | Standalone | Control flow                      |
| Pini et al. [50]      | 2015 | Event log, Process model    | Annotated process model (TS), Alignment matrix, Descriptive statistics | Alignment analysis | Healthcare | ProM | Control flow, Activity duration, Cycle time |
| Bolt et al. [8]       | 2015 | Event log                   | Annotated process model (TS)     | Transition system mining                       | Education           | RapidProM, RapidMiner | Control flow                      |
| Conforti et al. [13]  | 2015 | Event log                   | Annotated process model (C-BPMN) | C-BPMN mining, Log replay                     | Financial           | Apromore | Control flow, Cycle time          |
| Bee et al. [61]       | 2015 | Event log                   | Rule (as natural language statement) | Prime Event Structure, Partial Synchronized Product | Synthetic, Healthcare | Apromore | Control flow                      |
| Cuzzocrea et al. [17] | 2016 | Event log                   | Ensemble classifier              | Ensemble learning via stacking                 | Healthcare          | Weka, Standalone, ProM | Control flow, Cycle time          |
| Bolt et al. [10]      | 2016 | Event log                   | Annotated process model (TS)     | Transition system mining                       | Public administration | ProM | Control flow, Elapsed time        |
| Andrews et al. [2]    | 2016 | Event log                   | Annotated process model (C-BPMN) | C-BPMN mining, Log replay                     | Healthcare          | Apromore | Control flow, Cycle time          |
| Cuzzocrea et al. [18] | 2017 | Event log                   | Annotated process model (TS)     | Transition system mining, Clustering           | Logistic            | N/A       | Control flow, Cycle time          |
| Cuzzocrea et al. [19] | 2017 | Event log                   | Ensemble classifier              | Ensemble learning via stacking                 | Healthcare          | N/A       | Control flow, Cycle time          |
| Cuzzocrea et al. [20] | 2017 | Event log                   | Ensemble classifier              | Ensemble learning via stacking                 | Healthcare          | N/A       | Control flow, Cycle time          |
| Folino et al. [27]    | 2017 | Event log                   | Rule, Process model (TS)         | Clustering, Rule mining, Fuzzy mining          | Logistic            | N/A       | Control flow, Cycle time          |
| Wynn et al. [68]      | 2017 | Event log, Process model    | Annotated process model (PN), Alignment matrix, Descriptive statistics | Alignment analysis, Log replay                 | Insurance           | ProM | Control flow, Waiting time between activities |
| Gulden et al. [30]    | 2017 | Event log                   | Rhythm-eye view                  | Mining and configuring rhythm-eye visualization | Industrial          | Standalone | Control flow, Waiting time between activities |
| Ballambettu et al. [5]| 2017 | Event log                   | Annotated process model (TS)     | Process map mining                            | Logistic            | ProM | Control flow, Waiting time between activities |
| Folino et al. [26]    | 2018 | Event log                   | Ensemble classifier              | Ensemble learning via stacking                 | Healthcare          | Weka      | Control flow, Cycle time          |
| Bolt et al. [9]       | 2018 | Event log                   | Annotated process model (TS), Rule | Transition system mining, Decision tree induction | Public administration | ProM | Control flow, Elapsed time        |
| Nguyen et al. [47]    | 2018 | Event log                   | Matrix-based representation of Differential Graph | Perspective and Differential Graph              | Public administration, IcM | ProM, Apromore | Multiple perspectives |

Table 2. Overview and summary of the selected works
Another study that examines patient flow variations is presented by Suriadi et al. [57]. Patient flows include sequences of activities executed both in the Emergency Department (ED) and in the ward. The study aims at explaining event log variations across four different hospitals. To this end, the comparison of patient flows is done by discovering process models using the Fuzzy Miner [31], and the Heuristic Miner [66] using the four hospital sublogs. Then, a Petri net is derived from each discovered model, and its fitness is measured by aligning it with the process executions of the other sublogs (i.e., cross-validation) using the technique presented in [1]. Also, the authors conducted some descriptive analysis such as computing the maximum time for a patient to be discharged from ED across different hospitals to provide more insights about patient flow variations and the corresponding performance. Another work by Suriadi et al. [58] aims at improving the customer satisfaction of a company by reducing the processing time of its business processes. In particular, it tries to improve lengthy process executions, which, instead, are supposed to be fast and simple. The proposed approach employs a technique called Delta-Analysis. The same technique has been applied also in [49] to carry on Root Cause Analysis (RCA) for some specific process executions that take an unexpectedly long time to complete. RCA examines the existing causal relations between various factors that contribute to the execution time of a case via classification algorithms.

Pini et al. [50] apply some visualization techniques to tackle process variant analysis. The work provides a comparative process visualization technique to compare both performance and control flow of different process variants. The comparison is done using three perspectives, i.e., general model, superimposed model, and side-by-side comparison. Factors such as frequency of an activity and min/max/avg activity durations are used as objective measures to uncover differences among process variants. The general model perspective aims at emphasizing the performance differences among various process variants. The super imposed model perspective draws attention to process flows (i.e., activity ordering) by computing alignments [1]. The last perspective shows the waiting time between an activity and its successor, thus uncovering which activities inject delays in the whole process execution time. The work in [68] proposes an extension of the previous work by considering a normative process model alongside with event logs as inputs, and adding more data preparation facilities. It also provides comparative process visualizations at different levels-of-detail to improve interpretability for the end users.

The work in [2] employs a visualization and animation technique for highly varied patient flows, i.e., the systematic processing of a patient from arrival to discharge at a medical facility or emergency department. The objective is to shed light on the existing differences among patient flows. To this aim, the authors propose two techniques to capture both static and dynamic behavior in a set of process variants. The static view aims at highlighting control flow differences among process variants. To this end, a process model for each process variant is discovered and, then, a configurable process model is created by merging the discovered models [53]. The configurable model illustrates commonalities and variant-specific paths. The dynamic view is based on animating sublogs to highlight the differences in the executions of the variants, i.e., how cases in each variant flow through the models. Similarly to the mentioned work, the paper by Conforti et al. [15] presents guidelines and a set of handy and practical examples for the analysis of process variants. Here, a configurable model is created after removing process drift behavior from the event logs to obtain a stable process behavior for each process variant.

The work by Buijs et al. in [13] proposes a technique to identify the existing deviations between process models and the corresponding executions across various organizations. This work extends the approach proposed in [14] by explicit incorporating process models in the comparative analysis. Each process model is compared with the corresponding event log using the approach for computing alignments presented in [1]. The alignments show deviances between the models and the process
executions. In addition, cross-organizational process variants are compared using an alignment-matrix where columns and rows are process models and process variants, respectively. The matrix contains the fitness values computed by aligning each process variant against the process models.

The work by van Beest et al. [61] shows the behavioral distance between two sets of process executions. Behavioral differences are expressed using natural language statements highlighting exclusive frequent patterns in each set of process executions. The approach is based on encoding an event log as an annotated Event Structure [48]. In particular, a set of partially ordered runs (i.e., pairs of events that precede each other or are concurrent) are extracted from an event log. Each partially ordered run resembles a Prime Event Structure (PES), and the extracted set of partially runs shows causality relations. Also, a PES can be augmented with frequencies resulting in a Frequency-enhanced Prime Event Structure (FPES). The PESs of the process variants are compared by creating the Partial Synchronized Product (PSP) of the event structures [3]. The PSP shows which events can be executed synchronously in two event structures identifying a mismatch if this synchronous execution is not possible. The obtained mismatches are collected into a set of simple change patterns, which are subsequently translated into natural language statements [65].

Cordes et al. [16] present a visualization technique that compares process variants, which is independent of a specific process modeling language. In particular, a set of process models is discovered from a set of process variants and the comparison is done over the process models. In particular, the structure of two process models is compared in a similar way as in [41], i.e., by computing the minimum number of operations to transform one process model into another. The proposed algorithm compares the elements of two graphs and marks paired elements as unchanged, added, deleted, or changed to highlight the dissimilarities. Then, a view-model consistent with the input modeling language is generated for the end user. In the same vein, the work in [37] presents an approach independent of a specific process modeling language and based on directed graphs. The method provides some handy facilities to the end user to identify deviations. For example, the flow instance variations between two process variants can be seen in a single graph, or two process models can be compared for their structures using a difference graph model. An extension of this work is presented in [5], which compares process variants using Process Maps (annotated transition systems). In a first schema, a unified Process Map is generated by considering all process variants together. A second schema generates a difference Process Map including parts that are present in one process variant but not in the others. For common elements, pair-wise differences are computed to identify parts of the Process Map that are the most peculiar of a certain process variant.

The approach by Sun et al. [56] tackles the automatic evaluation of software processes. It assumes that two process variants are available, i.e., normal and anomalous executions. Process executions are encoded into a multidimensional space. The encoding schema is similar to the unigram encoding. The main idea is to infer from the two process variants a set of contrasting itemset patterns that do not share any features. If a new process execution contains all the features of a pattern, it can be classified as normal or anomalous. Similarly, Bose et al. [12] extract features such as Tandem Repeat and Maximal Repeat patterns [35] to encode traces into a multidimensional vector space. Then, association rule mining and decision tree induction techniques are used to extract rules characteristic of the process execution groups.

Swinnen et al. in [59] develop an approach to understand the reasons of variations in a procurement process. The proposed approach is unsupervised in the sense that process execution tags are unknown beforehand. A process model is discovered from an event log and is compared with a normative process model to uncover the differences. These differences then are used to group process executions. Then, association rule mining is used to extract rules from each group. Similarly, the work in [18] is also unsupervised. However, this work assumes that there are two pre-defined
process variants available. A model from the whole event log is discovered and is annotated with performance metrics for each process variant. Folino et al. [27] extended this work by identifying a set of rules to explain the differences between the two clusters of process executions.

The approach by Cuzzocrea et al. [17] adopts an ensemble learning schema to find a discriminating function that classifies process executions. The strategy is to encode a single process execution into a set of vector representations, i.e., to provide a multi-view schema of each process execution. After encoding an event log in this way, a base classifier is trained for every set of vector representations. Finally, the Stacking mechanism is used to perform the classification based on the outcomes of the base classifiers. This work was extended in [19, 20] by identifying the label of a process execution in a probabilistic way and by extracting rules to explain the discrepancies among process variants. A follow-up work by Folino et al. [26] proposes a peer-to-peer architecture for the discovery of base learners. The proposed architecture enables the business analyst to apply the approach in an online setting for a stream of traces. The stream of traces is processed by chunks thus allowing base learners to be adjusted periodically.

Bolt et al. [8] exploit Process Cubes [11] to split, group and compare process executions. Process cubes provide operations such as slice, dice, roll-up, and drill-down to break down process data and compare different groups or process variants to highlight dissimilarities. The work considers process executions containing the activities of a student. A process cube with various dimensions, such as “Course code”, “Grade” and “Activity Type” is created. The outcomes of this analysis are provided in different qualitative forms such as simple statistic values, dotted charts and comparisons of activity flows. A follow-up of this work is presented in [10]. Here, the differences between two sets of process executions are visualized by projecting them onto a transition system where states and transitions are colored to highlight the differences. The highlighted parts only show different dominant behaviors that are statistically significant, and rare differences are masked out for the sake of readability. The transition system is annotated with information such as the frequency of an event, the elapsed time of an event (i.e., the time elapsed between the beginning of the process execution and the occurrence of the event) for each process variant. This work was extended in [9] by inducting decision trees for each decision point (i.e., a node that branches) of the transition system. A set of rules is derived from the trained decision trees to explain the differences among process variants.

The work by Gulden et al. [30] proposes a circular time-line visualization, called rhythm-eye, to compare process executions in terms of execution time. In the proposed view, events are rendered as thin lines on top of the rhythm-eye ring. Average time values of each event type are represented by semi-transparent thicker circle segments, one per event type. Different event types are distinguished by colors. The approach computes a rhythm-eye view for each process variant and configures them to highlight differences.

Recently, Nguyen et al. [47] have proposed an approach to compare process variants via Perspective Graphs. A Perspective Graph is a graph-based abstraction of an event log where a node represents any entity referenced in an attribute of the event log (e.g., activity, resource, location), and an arc shows an arbitrary relation between entities. The approach starts by abstracting process executions in each process variant. The abstraction can be made on the order of activities or on any event attribute, e.g. the order in which resources hand over work to one another, or on a combination thereof (a schema). This results in a Perspective graph. The comparison can be done for any process perspective depending on the employed entities. To compare two Perspective Graphs a Differential Graph is computed. This graph contains common nodes and edges and also nodes and edges that appear in one perspective graph only. The weights of common nodes and edges are determined via statistical tests. Finally, the approach provides the identified differences in a matrix-based representation.
Primary study | Subsumed studies
---|---
Bolt et al. [9] | Bolt et al. [8, 10]
Ballambettu et al. [5] | Kriglstein et al. [37]
Cordes et al. [16] | 
Poelmans et al. [51] | 
Partington et al. [49] | Suriadi et al. [57, 58]
Swinnen et al. [59] | 
Sun et al. [56] | 
Bose et al. [12] | 
Wynn et al. [68] | Pini et al. [50], Andrews et al. [2], Conforti et al. [15]
Buijs et al. [13] | Buijs et al. [14]
van Beest et al. [61] | 
Folino et al. [26] | Cuzzocrea et al. [17–20], Folino et al. [27]
Lakshmanan et al. [40] | 
Gulden et al. [30] | 
Nguyen et al. [47] | 

Table 3. Primary and subsumed studies

**Primary and subsumed studies.** Among the papers that successfully passed both the inclusion and exclusion criteria, we determined primary studies that constitute an original contribution to process variant analysis and deviance mining, and subsumed studies that are similar to a primary study and do not provide a substantial contribution with respect to it. Specifically, a study is considered subsumed if:

- there exists a more recent and/or more extensive version of the study from the same authors (e.g., a conference paper is subsumed by an extended journal version), or
- it does not propose a substantial improvement/modification over a method that is documented in an earlier paper by other authors, or
- the main contribution of the paper is a case study or a tool implementation, rather than a new method, and the method is described and/or evaluated more extensively in more recent study by other authors.

As can be seen from the Table 3, a large number of works can be considered as a primary study because of the large variety of proposed techniques. We identified 15 primary and 14 subsumed studies.

### 4.2 Input data

As shown in Table 2, all the proposed approaches take as input an event log. The input event log may have a prior structure that can be used to identify process variants, or process variants can be created based on event attributes such as resources (see Figure 1). Some approaches also require a process model as input. In the following, we explain how the selected works employ input data in their analysis.

Some works assume that process executions are grouped or tagged beforehand. For example, Sun et al. [56] take as input two sets of software process executions, i.e., normal and anomalous executions. Suriadi et al. [57] use four groups of process executions coming from four different
hospitals. Similarly, the process variants in [2, 5, 9, 13, 14, 17, 19, 20, 26, 30, 61] are pre-defined. Although the input process executions in [47] are grouped beforehand, the approach can inherently create process variants based on performance attributes.

In other studies, process variants can be created based on performance data. The studies in [50, 68] use min/max/avg activity durations as objective measures to characterize different process variants. Cordes et al. [16], in their analysis, employ case attributes, such as the age or the region of a customer, to group together process executions. Likewise, Suriadi et al. [58] use the cycle time of a case to group process executions into cohorts. Bose et al. [12] group process executions of a process to repair malfunctions in X-ray machines according to the mean-time-to-repair of the parts that must be replaced. The work by Bolt et al. [8] uses Process Cubes [11] to group process executions based on performance data of students.

The works in [18, 27, 59] neither take as input a categorized set of process executions nor group them based on event or case attribute values. Indeed, the main aim of such studies is to discover process variants with no prior knowledge. However, the study in [18] assumes as prior knowledge the percentage of deviant and non-deviant cases.

Some approaches take as additional input a normative process model [13, 37, 68]. A normative process model is used as a reference model for quantifying to what extent the process variants differ from normative executions. The normative process model can be provided using different notations. The authors in [13] use BPMN, whereas the authors in [68] employ Petri nets. The approach presented in [37] does not pose any specific restrictions on the process modeling language employed, but for special concepts of certain languages developing extensions could become necessary.

4.3 Outcomes

The outputs of process variant analysis depend on the research questions and objectives considered in the different studies, and vary across different domains. However, as shown in Table 2, most of the works focus on providing explainable results showing how process variants differ from different perspectives. In particular, the outcomes of process variant analysis can be grouped based on the following categories:

- **Rule-based**: The works in [12, 27, 56, 58, 59] represent the existing discrepancies among process variants through a set of rules or causal relations. All these works provide the extracted rules according to different encoding schemas, but always as a conjunction of a set of antecedents, and a consequent that discriminates among different process variants. Similarly, the work in [51] finds itemsets, i.e., sets of activities, that differ for different process variants, whereas, the work in [40] finds frequent patterns characteristic of each process variant. Also, the analysis in [9] extracts a set of rules that can be used to assign a cohort label to each process execution. van Beest et al. [61] generate discriminative rules in terms of natural language statements.

- **Model-based**: A significant number of works provide as outputs process models that are easy to interpret for end users. The output is either a set of process models representing the behavior of the different process variants or an embodiment process model representing the behavior of an entire event log. The discovered process models are usually annotated with performance data. For example, the works in [8–10, 58] annotate the discovered transition systems with the frequency of the transitions between two states, whereas the approaches in [18, 27] annotate the discovered transition systems with performance data such as elapsed or remaining time. Andrews et al. [2] annotate the discovered configurable BPMN model with the length of stay of a patient in a hospital. The work in [49] generates BPMN models annotated with performance data and a transition system for each process variant highlighting
frequent paths. In the same way, Lakshmanan et al. [40] superimpose frequent patterns on the discovered transition systems. Suriadi et al. [57] derive Petri nets from the discovered transition systems representing the behavior of the process variants. Ballambettu et al. [5] use annotated transition systems to highlight differences among process variants. The approaches by Kriglstein et al. [37] and Cordes et al. [16] are more flexible and generate an annotated directed graph that can be translated into other notations. The study in [37] can also be provided with an input process model, which is annotated with the differences among the process variants. Similarly, Pini et al. [50] annotate an input transition system with various performance data such as the median execution times of activities. This work was extended by Wynn et al. in [68] where the input Petri net is projected into a flat model annotated with performance data such as waiting time between activities.

- **Descriptive**: Some works provide visual summaries and descriptive statistics for performance data or event attributes to highlight the differences among process variants. These outputs are standalone or can be integrated with the other outcomes, e.g., they can be used to annotate process models as mentioned earlier. Examples of techniques that use standalone descriptive statistics are [58] where basics statistics are used to identify which cases are more complex than others and Bolt et al. [8] that employ bar charts to show the number of students in each process variant and use dotted charts to visualize how many videos are watched by students in different cohorts. Poelmans et al. [51] identify process variants based on the patient’s length of stay in a hospital and then tabulate some important factors such as the number of cases and the average number of activities per case in each process variant. Also, several studies compare the control flow characteristics of process variants in tabular form [13, 14, 50, 57, 68]. The table includes fitness values showing how well a process execution from one cohort can be replayed by representative models of other process variants. Nguyen at al. [47] use a matrix-based structure for displaying statistically significant discrepancies among process variants derived from a Differential Graph. Gulden et al. [30] provide a circular visualization, called rhythm-eye, to compare the control flow structures of different process variants.

Some recent works provide a labeling or classifications of process executions. In these works, the outcome is a class label [17], or a set of probabilities that show how a process execution associates to different groups [19, 20, 26]. It is worth mentioning that the aim of these works is training a classifier for each process variant to label upcoming completed process executions. This approach is different from predictive process monitoring techniques, which predict the outcome of an ongoing process execution or estimate the required time to complete. Indeed, predictive process monitoring techniques operate in an online setting, whereas the mentioned studies operate in an offline setup.

### 4.4 Type of analysis

To conduct process variant analysis of process executions, different perspectives of the process under analysis can be taken into consideration. The process perspectives to look at in variant analysis depend on factors such as the research questions addressed and the availability of data. These perspectives also determine the type of outcome that needs to be produced and the underlying algorithms that need to be developed.

Based on the perspectives investigated, we can classify the types of analysis as:

- **Control flow**: In this type of analysis, a process execution is considered as an ordered set of activities discarding all available related contextual or performance attributes. Some of the studies that use the control flow perspective [9, 12, 27, 40, 51, 56, 58, 59, 61] generate a set of rules or patterns to express the control flow discrepancies in a set of process executions. Other works [2, 5, 8, 10, 16, 18, 37, 49, 50, 57, 68] extract process models from logs representing the
behaviors of different process variants. Some works provide a visual comparison to highlight discrepancies. For example, the work in [47] provides a compact matrix-based representation of statistically significant differences from a Differential Graph. Similarly, Gulden et al. [30] produces rhythm-eye views to compare process variants based on control flow. Finally, several studies [13, 14, 50, 57, 68] compare the control flow characteristics of process variants using alignments.

• **Performance analysis:** Recent works have focused more on the analysis of contextual or performance attributes. This perspective is important since a set of process executions with the same control flow could have different cycle times or use different types of resources. Most of these works consider time-related performance data in their analysis. For example, Poelmans et al. [51] consider the length of stay of a patient for cycle time analysis to discover discrepancies among patients with the same control flow structures. In [2, 18, 27, 49, 50, 57], the authors take into account the cycle time of process executions to separate process executions into groups and then find control flow characteristics of slow cases. In the same vein, the work by Nguyen et al. [47] discovers a control flow model for any combination of time-based attribute values. The studies in [5, 30, 49, 57, 68] work with the waiting times between activities across different process variants to understand the existing performance variations, whereas Pini et al. [50] consider the median duration of each activity. Bolt et al. [9, 10] investigate the elapsed time, i.e., the time between the starting point of a process execution and the occurrence of a certain event to identify performance deviations. Other works start from pre-defined groups of process executions and leverage both control flow and performance data to characterize those groups. For example, in [17, 19, 20, 26], the authors use both control flow and cycle time of process executions to train an ensemble classifier. The classifier assigns an upcoming process execution to a process variant.

Except the work in [47], it is interesting to observe that none of the techniques we retrieved considers the possibility of comparing process variants along other perspectives besides the above two. Yet, it is conceivable that two process variants may differ along the resource perspective (e.g. different resource pool), or along the data perspective.

### 4.5 Family of algorithms

When conducting process variant analysis, the underlying algorithms used are strongly influenced by the input data and by the accessibility of performance attributes. Nevertheless, the proposed algorithms share the ability of providing explainable results. Broadly speaking, the algorithms used in the selected papers belong to two main families:

• **Process mining:** This family of algorithms uses process mining techniques to uncover differences among process variants. The majority of the proposed approaches discover a process model for every process variant, and then compare them to highlight the differences. In [8, 10, 58], the authors discover an annotated transition system where states and transitions are colored to show different dominant behaviors (representing different process variants) that are statistically significant. Factors such as frequency and elapsed time of an event are considered in the analysis. Similarly, in [18], a transition system is discovered from the whole event log and annotated with performance metrics characterizing each process variant. Kriglstein et al. [37] compute a directed graph, called Difference model, to highlight the existing differences between two process variants. In this work, a normative process model representing the expected process behavior can be provided as input. This work was extended by Ballambettu et al. [5] where annotated transition systems are used (called process maps) to represent the behaviors of different process variants. Suriadi et al. [57] discover a Petri
net for every process variant and quantify the closeness of their control flow structures using alignments [1]. In particular, alignments provide a fitness value that tells how good a process model discovered for a process variant can replay the observed executions available in the logs corresponding to the other cohorts. In the same way, the works in [13, 14, 50, 68] compute alignments to come across the existing control flow differences among process variants. Most of these works take as input a normative model representing the expected behavior of the process. The analysis by Partington et al. [49] first discovers a process model using the Fuzzy Miner from the whole event log, then it replays process executions of different process variants on the discovered model to characterize them using infrequent-traversed paths. Cordes et al. [16] compare the structures of two process models discovered from two process variants using TGraphs [24]. A TGraph is an intermediate representation of a process model, wherein no distinction among different types of nodes and different types of edges is assumed. Each node and edge, however, carries additional information to preserve the semantics of the original process model. For example, in a Petri net, a node can be marked as a transition or a place. Two TGraphs are compared using the Snapshot-diff algorithm [39], which produces a Difference model. In particular, the algorithm compares two graphs by comparing their elements and marking them as unchanged, added, deleted, or changed to highlight dissimilarities. van Beest et al. [61] encode an event log as an annotated event structure [48], which is a directed acyclic graph where nodes represent event occurrences sharing a common history. Annotated event structures also keep information about the frequency of each event. The technique extracts a set of partially ordered runs where pairs of events can precede each other or be concurrent. Each partially ordered run resembles a prime event structure, i.e., a graph of events representing the causal relations between events. The partially ordered runs are merged to derive a prime event structure of the full log. When different logs corresponding to different process variants are available different prime event structures are derived using the above procedure and then compared using the partial synchronized product of the event structures [3]. The identified mismatches are collected into a set of simple change patterns, which are subsequently translated into natural language statements [65]. Andrews et al. [2] discover a BPMN process model for each process variant, and then build a configurable process model obtained by merging the discovered models using the technique proposed in [53]. The configurable model illustrates commonalities and variant-specific paths. The paper also proposes a log replaying technique using a heuristic-based backtracking algorithm to compare a process execution and a BPMN model. Nguyen et al. [47] discover, from the process executions corresponding to a process variant, a perspective graph taking into consideration control flow and different combinations of performance attributes. Then, two perspective graphs are compared and merged into a Differential Graph in which the elements that are statistically different in the perspective graphs are highlighted. Finally, Poelmans et al. [51] use the formal concept analysis [28] to capture a representative set of activities in each process variant. Formal concept analysis is a method for deriving implicit relationships between objects (in process variant analysis activities) described through a set of attributes.

**Machine learning:** This family of algorithms exploits machine learning or statistical algorithms to analyze process variants. Sun et al. [56] use contrast itemsets [6] to characterize process variants. Contrast itemsets are composed of attribute values that differ across groups of process executions. Bose et al. [12] transform process executions into multidimensional vector representations using as features frequent control flow patterns. Then, they apply association rule mining and decision tree induction to infer a set of rules that characterize process variants. Lakshmanan et al. [40] find frequent sequence patterns using Sequential
Pattern Mining with bitmap representation (SPAM) [4]. The patterns are used to represent every process execution as a Bag-of-Pattern (BoP). Then, Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [25] is used to cluster process executions in different cohorts. The work in [17], after transforming process executions into multidimensional feature vectors, adopts an ensemble method (Bayesian Model Averaging) to learn a classifier via stacking [22]. Stacking is a meta-learning task in machine learning where a classifier uses the output of other base classifiers to better classify or label a process execution. In particular, meta-learning allows a learner to not only learn from historical data, but also from other learning tasks. The approaches in [19, 20] extend the previous work by adopting the Hidden Naive Bayes classifier [69] at the meta-learning level. This type of classifier provides probabilistic outcomes. Folino et al. [26] extend the previous works by proposing a peer-to-peer computing architecture to speed up the training phase of base learners.

It is worth pointing out that, though we broke up the process variant analysis algorithms into two families, some works belong to both. For example, Swinnen et al. [59] first discover a process model using the Fuzzy Miner, and then find discrepancies between the discovered model and a normative model to assign process executions to different process variants. Then, the authors use the Apriori algorithm [54] from association rule mining to find a set of rules characterizing each process variant. Similarly, Folino et al. [27] propose an iterative optimization algorithm to infer a set of rules to group process executions into process variants. Then, a process model is discovered for each cohort using the Fuzzy Miner. Works that are in between the two families are the one presented in [58] that infers a set of causal relation rules to characterize lengthy process executions and the analysis presented in [57] that uses K-means clustering to group the input set of process executions. Finally, Bolt et al. [9] also use a typical process mining algorithm to create an annotated transition system starting from a log, and then, for every decision point in the transition system, train a classifier to distinguish different process variants.

4.6 Evaluation data and application domain

As reported in Table 2, most of the surveyed methods have been validated on at least one real-life event log, and a few studies were additionally validated on simulated (synthetic) logs. Most of the real-life logs employed are publicly available in the 4TU Center for Research Data\footnote{https://data.4tu.nl/repository/collection:event_logs_real}. Among the methods that use real-life logs, we observed a growing trend to use publicly available datasets, as opposed to private logs that hinder the reproducibility of the results.

Process variant analysis is attractive and beneficial in domains where a single process model is executed across different organizations. A good example is provided by SaaS applications, where a single version of an application, with a single configuration, is used for different customers, such as applications for logistics, Incidence Management (IcM), financial management and healthcare management. From Table 2, we notice that most of the selected works pertain to healthcare (12 studies), logistics (4 studies), public administration (5 studies), industrial and insurance organizations (5 studies), financial institutions (3 studies), education systems (1 study) and IcM systems (1 study).

4.7 Implementation

Providing publicly available implementations and experimental data facilitates the reproducibility of the results and enables researchers to build on past works. According to Table 2, around half of the methods provide an implementation as a plug-in of the process mining tools ProM [52] and Apromore [38]. Both the aforementioned frameworks are open-source and portable, which allows
Table 4. Encoding process executions using unigram

| Case_id | City | Sex | Product | Cycle time (s) | Order | Pay in cash | Pay by card | Approval | Disapproval |
|---------|------|-----|---------|---------------|-------|-------------|-------------|----------|-------------|
| 1       | NY   | M   | Book    | 183           | 1     | 1           | 0           | 1        | 0           |
| 2       | MA   | F   | Sofa    | 58,690        | 1     | 1           | 1           | 1        | 1           |
| 3       | LA   | M   | T.V.    | 960           | 1     | 0           | 1           | 1        | 0           |
| 4       | LA   | F   | Book    | 960           | 1     | 1           | 0           | 1        | 0           |

Table 5. Encoding process executions using bigram

| Case_id | City | Sex | Product | Cycle time | Order | Order | Order | Pay in cash | Pay by card | Pay in cash | Pay by card | Pay in cash | Pay by card | Pay in cash | Pay by card | Pay in cash | Pay by card | Pay by card | Pay by card |
|---------|------|-----|---------|------------|-------|-------|-------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| 1       | NY   | M   | Book    | 183        | 1     | 1     | 0     | 1           | 0           | 0           | 0           | 0           | 0           | 0           | 0           | 0           | 0           |
| 2       | MA   | F   | Sofa    | 58,690     | 0     | 0     | 1     | 1           | 1           | 1           | 1           | 0           | 1           | 0           | 0           | 0           | 0           |
| 3       | LA   | M   | T.V.    | 960        | 0     | 1     | 0     | 0           | 0           | 0           | 0           | 0           | 1           | 0           | 0           | 0           | 0           |
| 4       | LA   | F   | Book    | 960        | 1     | 0     | 0     | 1           | 0           | 0           | 0           | 0           | 0           | 0           | 0           | 0           | 0           |

researchers to easily develop and test new algorithms. Similarly, a few works employ other tools such as Disco⁴, Nitro⁵, and RapidProM⁶.

Several techniques that employ machine learning algorithms use Weka [33], which is an open-source library implementing machine learning algorithms. Other machine-learning-based approaches use the Hidden Markov Model toolbox for Matlab⁷ and RapidMiner [34].

Finally, some works implemented their methods as standalone applications; others did not provide any prototype at all, or provided only parts of them.

5 UNIFYING FRAMEWORK

As outlined in the previous section, a wide range of methods have been proposed to tackle the problem of process variant analysis. However, because of the heterogeneous nature of the underlying algorithms, their inputs, and their outputs, the classification proposed in the previous section, while comprehensive, does not provide us with a unifying view of the state of the art in the field.

As a first step towards building a unifying view of the field, we propose an alternative classification of existing methods based on the observation that some of the methods seek to identify discriminating characteristics or patterns, while other approaches discover a model of each of the variants and then compare the variants based on the discovered models. This observation leads us to classify existing approaches into three categories: discriminative, generative and hybrid. This broad classification is a step towards unifying the various strands of research in the field, by bringing them together in terms of their underpinning paradigms. Below, we provide a detailed explanation of each of these three categories.

5.1 Discriminative

A discriminative approach to process variant analysis leverages techniques that aim at identifying features or patterns that can be extracted from process executions directly to discriminate among process variants and highlights the existing differences. These features include both control flow features and performance attributes and can range from frequency of individual activities/attributes, itemsets of activities/attributes, prefixes of process executions or their subsequences or, a combination of them.

⁴https://fluxicon.com/disco/
⁵https://fluxicon.com/nitro/
⁶http://rapidprom.org/
⁷https://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html
In general, these approaches can use two mechanisms to infer discriminatory features:

- **Vector-based**: This mechanism encodes every process execution into a vector representation labeled either with the corresponding process variant (to discriminate among different cohorts) or with a performance attribute (to discriminate among different values of this attribute within the same cohort). Then, a classifier is trained using these multidimensional representations of process executions. The trained model aims at identifying which dimension or combination of dimensions of the input vectors better contribute to the determination of the label. The crucial part of this mechanism is that a process execution is encoded into a vector representation, i.e., $g: \sigma \rightarrow \mathbf{x}$. There are several techniques that use this mechanism, though most of them use lossy encodings. A lossy encoding does not capture the entire information of a process execution when it is transformed into a feature vector, thus some information can be lost during the transformation. One easy way for implementing a lossy encoding is by using n-grams. An n-gram is a sequence of n items. For example, for the event log presented in Table 1, the corresponding unigram and bigram representations alongside with performance attributes are presented in Tables 4 and 5. It is easy to see why an n-gram is a lossy encoding. Indeed, the unigram encoding in Table 4 ignores the existing order of activities in the process executions, i.e., it considers a process execution as a bag of activities. It is clear that the n-gram encoding for $n \geq 2$ better captures the activity orders although it increases the curse of dimensionality. Folino et al. [26] employ n-gram encodings for $n \in [1, 4]$ to examine which patterns better contribute to the prediction of cohorts. It is noteworthy that there are several more sophisticated encoding schemas that better capture the behavior observed in process executions. For example, Sun et al. [56] use a modified version of unigrams where the position of an element is also considered as a dimension, whereas Bose et al. [12] use tandem repeats and maximal repeats patterns to encode traces into multidimensional vectors. In their works, Cuzzocrea et al. [17, 19, 20] consider different combinations of these patterns. Similarly, Nguyen et al. [46] extensively experimented different encoding schemas using different combinations of features. The classifiers used to classify feature vectors (see Figure 4) range from rule-based classifiers (high explainability, low accuracy) to ensemble learning algorithms (low explainability, high accuracy).

- **Model-based**: This mechanism uses an input process model, and considers it as the normative behavior. The main idea is to determine whether the observed behavior in each process variant, i.e., a process execution, agrees with the expected behavior or not. To implement this mechanism, two similar techniques can be used, namely alignment analysis and log replay. Although computing alignments is optimal in finding deviations, its complexity is exponential.
Fig. 7. Merging two discovered process models into one

[1]. Therefore, log replaying methods, having a lower complexity, can be leveraged to identify deviations. Using log replay, it is possible to monitor the frequency of every process path observed in an event log. Thus, frequent and infrequent paths can be determined and used to highlight the discrepancies among the behaviors of process variants and between the behavior of each process variant and the normative model. Figure 6 shows the frequencies of paths on an input process model after replaying the process executions of two different process variants on the model. The thickness of an edge shows how many times the edge has been traversed by process executions in a process variant so that it is easy to extract frequent paths characteristic of each process variant.

5.2 Generative

A generative approach to process variant analysis leverages process model comparison techniques to shed light on existing differences among process variants. These approaches, usually, do not represent discrepancies in terms of patterns or rules as in the descriptive approach; instead, they present discrepancies graphically.

In general, a generative approach is composed of two stages. In the first stage, a process model for every process variant is discovered. The discovered model can be represented using different formalisms such as Petri nets, transition systems, BPMN models, Hidden Markov Models. In the second stage, the discovered models from each process variant are compared with each other or with a normative process model. In most of the cases, the discovered process models are merged into a single model where the behaviors of the single cohorts are highlighted [2, 9, 10, 16, 37].

There are several sophisticated methods for merging process models that are beneficial for generative approaches [53]. However, a straightforward way for merging two process models representing two process variants was presented in [5, 37] and is illustrated in Figures 7 (a), (b) and (c). Figures 7 (a), (b) show the control flow structures and the corresponding path frequencies of two different process variants. To have the representation of both behaviors in a single process model, the two models can be merged as shown in Figure 7(c).

The main advantage of generative approaches over discriminative approaches is not only related to the readability of the results that are easier to understand for end users, but also to their lower sensitivity to noise. This is due to the fact that the process discovery techniques used in these approaches can be seen as a filtering or pre-processing step that pull out noise or unusual behaviors before identifying the discrepancies, which leads to having more comprehensible results.
5.3 Hybrid

Hybrid approaches are a combination of a generative phase and a discriminative phase. The idea behind hybrid approaches is to discover discriminatory patterns or rules and project them onto a process model. These approaches are usually composed of several stages. Usually a hybrid approach starts by discovering a process model from the log corresponding to a process variant and, then, discriminative patterns are discovered to characterize different process variants. Finally, the discriminative patterns are superimposed on the discovered model to highlight discriminative parts. For example, the approach presented in [40] finds frequent sequence patterns using Sequential Pattern Mining and project them onto the process model discovered from the process executions of each process variant.

One of the most straightforward hybrid approaches consists in discovering a process model for every process variant and then applying cross-validation. Usually, cross-validation is accomplished by computing alignments to quantify how similar control flow structures are across different process variants [13, 50, 57]. For example, assume that there are \( m \) process variants, and \( m \) process models representing them are discovered. Then, a process variant is selected, and its process executions are aligned with the other \( m - 1 \) process models. The procedure repeats for all process variants. The result is a matrix structure containing the average fitness values, which show the similarities in terms of control flow of the different process variants.

6 Conclusion

Understanding the differences between multiple process variants can help analysts and managers to make informed decisions as to how to standardize or otherwise improve a business process, for example by helping them find out what factors lead to a given variant exhibiting better performance than another one. Various methods for process variant analysis based on event logs have been proposed in the past decade. However, to this date, the field remains rather fragmented.

As a first step towards building up a unified view of the field, this article provided a survey and a classification of existing methods for business process variant analysis. The relevant studies were identified through a systematic literature review, which retrieved 29 studies. Out of these 29 studies, 15 of them propose distinct methods (primary studies). Through further analysis of the primary studies, a taxonomy was proposed based on four aspects: (1) the type of input data required; (2) the provided outputs; (3) the type of analysis, and (4) the algorithms employed. While analyzing the algorithms employed, we noticed that some of the methods rely on the identification of characteristics or patterns that are frequently present in one variant and not in the other variants (discriminative approaches). Other approaches, in contrast, seek to discover a model for each of the process variants and then compare the discovered models (generative approaches). It was found that 8 out of the 15 primary studies employ a generative approach, while the remaining 9 employ discriminative or hybrid generative-discriminative approaches.

The study shed light into research gaps in the field and corresponding avenues for future work. First, most of the studies consider time-related performance, thus ignoring other performance dimensions such as cost, quality, flexibility, or compliance. Second, while a large subset of existing approaches focus on control-flow differences, the question of comparing process variants along the data perspective or the resource perspective has not received attention. Finally, most of the proposed approaches show the identified deviations in a descriptive way, without backing up the detected differences between process variants with statistical tests or causal analysis, which could help to generate recommendations for addressing deficiencies in one or more of the analyzed process variants. In other words, this study calls for the development of multi-perspective approaches to process variant analysis, which would seek not only to identify differences between two or more
variants, but also, to conclusively determine which of these differences contribute to observed differences in the performance of the process variants. Such multi-perspective and statistically grounded approaches could help analysts and managers to obtain insights into how to improve the performance of specific variants of a business process.

ACKNOWLEDGMENTS

This research is partly funded by the Australian Research Council (grant DP180102839) and the European Research Council (PIX project).

REFERENCES

[1] A. Adriansyah. 2014. Aligning observed and modeled behavior. Ph.D. Dissertation. Department of Mathematics and Computer Science.

[2] Robert Andrews, Suriadi Suriadi, Moe Thandar Wynn, Arthur H. M. ter Hofstede, Anastasia Pika, Huang Huy Nguyen, and Marcello La Rosa. 2016. Comparing static and dynamic aspects of patient flows via process model visualisations. Science & Engineering Faculty (2016).

[3] Abel Armas-Cervantes, Paolo Baldan, Marlon Dumas, and Luciano García-Bañuelos. 2014. Behavioral Comparison of Process Models Based on Canonically Reduced Event Structures. In Business Process Management, Shazia Sadiq, Pnina Soffer, and Hagen Völzer (Eds.). Springer International Publishing, Cham, 267–282.

[4] Jay Ayres, Jason Flannick, Johannes Gehrke, and Tomi Yu. 2002. Sequential PAttern Mining Using a Bitmap Represen-
tation. In Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD ’02). ACM, New York, NY, USA, 429–435. https://doi.org/10.1145/775047.775109

[5] Nithish Pai Ballambettu, Mahima Agumbe Suresh, and R. P. Jagadeesh Chandra Bose. 2017. Analyzing Process Variants to Understand Differences in Key Performance Indices. In Advanced Information Systems Engineering, Eric Dubois and Klaus Pohl (Eds.). Springer International Publishing, Cham, 298–313.

[6] Stephen D. Bay and Michael J. Pazzani. 2001. Detecting Group Differences: Mining Contrast Sets. Data Mining and Knowledge Discovery 5, 3 (01 Jul 2001), 213–246. https://doi.org/10.1023/A:1011429418057

[7] Patrick Biernacki and Dan Waldorf. 1981. Snowball Sampling: Problems and Techniques of Chain Referral Sampling. Sociological Methods & amp Research 10 (11 1981).

[8] Alfredo Bolt, Massimiliano De Leoni, Wil M.P. van der Aalst, and Pierre Gorissen. 2015. Exploiting process cubes, analytic workflows and process mining for business process reporting: A case study in education. In Data-Driven Process Discovery and Analysis (SIMPDA 2015), December 9-11, 2015, Vienna, Austria (CEUR Workshop Proceedings), P. Ceravolo and S. Rinderle-Ma (Eds.). CEUR-WS.org, 33–47.

[9] Alfredo Bolt, Massimiliano De Leoni, and Wil M.P. van der Aalst. 2018. Process variant comparison: Using event logs to detect differences in behavior and business rules. Information Systems 74, Fundamenta Informaticae 94 3-4 2009 (2018), 53–66. https://doi.org/10.1016/j.is.2017.12.006

[10] Alfredo Bolt, Massimiliano de Leoni, and Wil M. P. van der Aalst. 2016. A Visual Approach to Spot Statistically-
Significant Differences in Event Logs Based on Process Metrics. In Advanced Information Systems Engineering, Selmin Nurcan, Pnina Soffer, Marko Bajec, and Johann Eder (Eds.). Springer International Publishing, Cham, 151–166.

[11] Alfredo Bolt and Wil M. P. van der Aalst. 2015. Multidimensional Process Mining Using Process Cubes. In Enterprise, Business-Process and Information Systems Modeling, Khaled Gaaloul, Rainer Schmidt, Selmin Nurcan, Sérgio Guerreiro, and Qin Ma (Eds.). Springer International Publishing, Cham, 102–116.

[12] R. P. J. C. Bose and W. M. P. van der Aalst. 2013. Discovering signature patterns from event logs. In 2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM). 111–118.

[13] Joos C. A. M. Buijs and Hajo A. Reijers. 2014. Comparing Business Process Variants Using Models and Event Logs. In Enterprise, Business-Process and Information Systems Modeling, Ilia Bider, Khaled Gaaloul, John Krogstie, Selmin Nurcan, Henderik A. Proper, Rainer Schmidt, and Pnina Soffer (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 154–168.

[14] J. C. A. M. Buijs, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. 2012. Towards Cross-Organizational Process Mining in Collections of Process Models and Their Executions. In Business Process Management Workshops, Florian Daniel, Kamel Barkaoui, and Schahram Dustdar (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 2–13.

[15] Raffaele Conforti, Marlon Dumas, Marcello La Rosa, Abderrahmane Maaradji, Hoang Nguyen, Alireza Ostovar, and S Raboczi. 2015. Analysis of business process variants in apromore. CEUR Workshop Proceedings 1418 (01 2015), 16–20.

[16] Carsten Cordes, Thomas Vogelgesang, and Hans-Jürgen Appelrath. 2015. A Generic Approach for Calculating and Visualizing Differences Between Process Models in Multidimensional Process Mining. In Business Process Management Workshops, Fabiana Fournier and Jan Mendling (Eds.). Springer International Publishing, Cham, 383–394.
[63] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. 2003. Workflow Patterns. *Distributed and Parallel Databases* 14, 1 (01 Jul 2003), 5–51.

[64] Ilya Verenich, Marlon Dumas, Marcello La Rosa, Fabrizio Maria Maggi, and Irene Teinemaa. 2018. Survey and cross-benchmark comparison of remaining time prediction methods in business process monitoring. *CoRR* abs/1805.02896 (2018).

[65] Barbara Weber, Manfred Reichert, and Stefanie Rinderle-Ma. 2008. Change Patterns and Change Support Features - Enhancing Flexibility in Process-aware Information Systems. *Data Knowl. Eng.* 66, 3 (Sept. 2008), 438–466.

[66] A Weijters, Wil M. P. Aalst, and Alves A K Medeiros. 2006. *Process Mining with the Heuristics Miner-algorithm*. Vol. 166.

[67] D. H. Wolpert and W. G. Macready. 1997. No Free Lunch Theorems for Optimization. *Trans. Evol. Comp* 1, 1 (April 1997), 67–82. https://doi.org/10.1109/4235.585893

[68] M.T. Wynn, E. Poppe, J. Xu, A.H.M. ter Hofstede, R. Brown, A. Pini, and W.M.P. van der Aalst. 2017. ProcessProfiler3D: A visualisation framework for log-based process performance comparison. *Decision Support Systems* 100 (2017), 93 – 108. Smart Business Process Management.

[69] Harry Zhang, Liangxiao Jiang, and Jiang Su. 2005. Hidden Naive Bayes. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 2 (AAAI’05)*. AAAI Press, 919–924. http://dl.acm.org/citation.cfm?id=1619410.1619480