Information Systems 37 (2012) 654–676

Contents lists available at SciVerse ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs

Jochen De Weerdt [a,*], Manu De Backer [a,b], Jan Vanthienen [a], Bart Baesens [a,c]

[a] *Department of Decision Sciences and Information Management, Katholieke Universiteit Leuven, Naamsestraat 69, B-3000 Leuven, Belgium*
[b] *Department of Business Administration and Public Management, Hogeschool Gent, Universiteit Gent, Voskenslaan 270, B-9000 Ghent, Belgium*
[c] *School of Management, University of Southampton, Highfield Southampton SO17 1BJ, United Kingdom*

A R T I C L E I N F O

Article history:
Received 6 April 2011
Received in revised form
18 November 2011
Accepted 28 February 2012
Recommended by: M. Weske
Available online 5 March 2012

Keywords:
Process mining
Benchmarking
Real-life event logs
Accuracy
Comprehensibility

A B S T R A C T

Process mining is the research domain that is dedicated to the a posteriori analysis of business process executions. The techniques developed within this research area are specifically designed to provide profound insight by exploiting the untapped reservoir of knowledge that resides within event logs of information systems. Process discovery is one specific subdomain of process mining that entails the discovery of control-flow models from such event logs. Assessing the quality of discovered process models is an essential element, both for conducting process mining research as well as for the use of process mining in practice. In this paper, a multi-dimensional quality assessment is presented in order to comprehensively evaluate process discovery techniques. In contrast to previous studies, the major contribution of this paper is the use of eight real-life event logs. For instance, we show that evaluation based on real-life event logs significantly differs from the traditional approach to assess process discovery techniques using artificial event logs. In addition, we provide an extensive overview of available process discovery techniques and we describe how discovered process models can be assessed regarding both accuracy and comprehensibility. The results of our study indicate that the HeuristicsMiner algorithm is especially suited in a real-life setting. However, it is also shown that, particularly for highly complex event logs, knowledge discovery from such data sets can become a major problem for traditional process discovery techniques.

1. Introduction

In today's service organizations, business processes generate large amounts of data. Event logs of information systems are such data sources that contain an untapped reservoir of knowledge about the way employees conduct every-day business transactions. Event logs essentially capture what business activities happened at a certain moment in time. Hence, analyzing these event logs is a promising way of acquiring insights into the semantics of the real business processes.

The research domain that is concerned with knowledge discovery from event logs is called process mining [1,2]. It can be situated at the intersection of the fields of data mining and Business Process Management (BPM). Process mining is related to data mining [3] and to the more general domain of knowledge discovery in databases (KDD) because of the common goal of learning from large data repositories. In this way, researchers within the field have developed numerous quantitative techniques and approaches that allow to examine the real execution traces of business activities. Similarly, process mining can

* Corresponding author. Tel.: +32 16 32 68 87; fax: +32 16 32 66 24.
E-mail addresses: jochen.deweerdt@econ.kuleuven.be (J. De Weerdt), manu.debacker@econ.kuleuven.be (M. De Backer), jan.vanthienen@econ.kuleuven.be (J. Vanthienen), bart.baesens@econ.kuleuven.be (B. Baesens).

0306-4379/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.is.2012.02.004

be associated with the field of Business Process Management as one of its major objectives is gaining insight into business processes. As a result, process mining fits flawlessly into the BPM life cycle framework [4].

Furthermore, we point to an important terminology issue concerning process mining and process discovery. Process mining describes a family of a posteriori analysis techniques for extracting knowledge from event logs, whereas process discovery only deals with extracting control-flow models. In spite of the broadness of the process mining field (e.g. [5,6]), most of the attention in the process mining literature has been given to process discovery techniques. Also this study fits within the narrower scope of process discovery.

Process mining techniques are specifically designed to provide insights into the real operational semantics of business processes. The distinctive, analytical focus on business processes makes process mining a research field on its own. The difference between process mining and traditional business intelligence (BI) tools is eminent. The added value of process mining over BI and other OLAP reporting tools lies in the depth of the analysis as BI tools mainly focus on the display of Key Performance Indicators (KPIs). The major drawback of classical BI tools is that they lack the ability to provide thorough insight into the root causes of process inefficiency and erroneousness. In contrast, existing and novel process mining techniques are an ideal means to obtain this profound insight.

One of the major driving forces of the success of process mining is definitely the graphical visualization. However, knowledge discovery in real-life event logs faces the crucial requirement to extract useful, interpretable information, preferably in line with expectations of domain specialists. Research with respect to the application of process mining techniques in real-life settings has indicated that the currently available techniques often suffer from incomprehensibility.

With this study, we want to thoroughly assess the quality of existing process discovery techniques. As such, the major contributions of this study are

- a comprehensive overview of process discovery techniques,
- the first comparative benchmarking study of process discovery techniques using multiple real-life event logs,
- a multi-dimensional evaluation study focussing on both accuracy and comprehensibility,
- demonstration of the F-score as an evaluation approach for combining accuracy dimensions.

The assessment of process discovery techniques centers on accuracy and comprehensibility, as depicted in Fig. 1. The former refers to how well a process discovery technique is able to render process models that capture the behavior in an event log soundly, hereby balancing between overly general or overly precise models. In contrast, comprehensibility entails the assessment of the understandability of the discovered process models in terms of their complexity and ease of interpretation.

The paper is organized as follows. The next section provides a comprehensive literature review of process discovery techniques. Section 3 presents a discussion on how to evaluate process discovery techniques with respect to accuracy. Furthermore, in Section 4, we outline a number of process model complexity metrics in order to assess the comprehensibility of discovered process models. Then, Section 5 presents the empirical setup of the benchmarking experiment before the results are discussed in Section 6. The last sections are devoted to a discussion on the aspect of representational bias and conclusions.

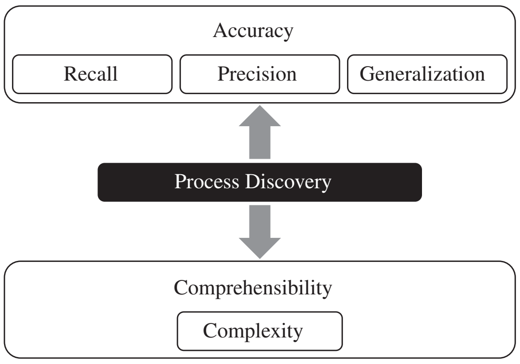

Fig. 1. Overview of the quality dimensions of a discovered process model.

2. Discussion of process discovery techniques

In the past 12 years, many process discovery techniques have been proposed (see Table 1). Algorithmic, machine learning as well as probabilistic approaches have been conceived. One of the major drivers for process miners is the development of the ProM framework [7]. This supporting framework developed at TU/Eindhoven allows the development of all kind of process intelligence techniques. One of its most important features is the underlying file format for process event logs, formerly known as MXML, but recently improved to XES (eXtensible Event Stream).

2.1. Early approaches

The foundational approaches to process discovery were formulated by Agrawal [8], Cook and Wolf [9], and Datta [10]. Cook and Wolf proposed three different approaches in the context of software engineering. Their RNet, Ktail and Markov methods approached the discovery of models from event-based data from a statistical, algorithmic and probabilistic viewpoint. Nonetheless, the idea of applying process discovery in the field of workflow management systems stems from Agrawal et al. and Datta.

Manilla and Meek [11] describe a technique to learn two-component mixture models of global partial orders that provide an understandable, global view of a set of sequences. As a result, the technique cannot cope with concurrency and many other typical problems in the field of process mining. Another process mining tool was developed by Schimm [12,13]. His Process Miner is entirely based on data mining techniques and is able to discover complete and minimal process schemes from event-based data. Yet, Process Miner

Table 1
Overview of process discovery techniques.

Name	Author(s)	Year	Typical problems					Implemented in ProM	Approach						Assessed in this study
			Noise	Duplicate tasks	Hidden tasks	Non-free choice constructs	Loops		Genetic algorithm	Classification-based	Clustering-based	Algorithmic approach	Inductive logic progr.	Probabilistic approach	
General DAG	Agrawal et al. [8]	1998	✓		✓							✓			
B-F(k,c)-algorithm	Datta [10]	1998	✓				✓					✓			
Rnet, Ktail, Markov	Cook and Wolf [9]	1998	✓		✓		✓			✓		✓		✓	
Global partial orders	Manilla and Meek [11]	2000	✓					✓				✓		✓	
Process Miner	Schimm [12,13]	2002		✓			✓			✓	✓				
α/α^{+}	van der Aalst et al. [14,2]	2004						✓				✓			✓
InWoLvE—splitpar	Herbst and Karagiannis [30]	2004	✓	✓	✓							✓		✓	
Multi-phase Miner	van Dongen and van der Aalst [31]	2005						✓				✓			
Workflow Miner	Gaaloul et al. [32]	2005						✓				✓			
HeuristicsMiner	Weijters et al. [33,17]	2006	✓		✓	✓	✓	✓				✓			✓
DWS Mining	Greco et al. [25]	2006	✓			✓		✓		✓	✓				
Rule-based approach	Maruster et al. [19]	2006	✓							✓					
–	Ferreira and Ferreira [20]	2006											✓		
Genetic Miner	Alves de Medeiros et al. [22]	2007	✓		✓	✓	✓	✓	✓						✓
DT Genetic Miner	Alves de Medeiros et al. [34]	2007	✓	✓	✓	✓	✓	✓	✓						✓
Fuzzy Miner	Günther and van der Aalst [18]	2007	✓				✓	✓			✓	✓			
α^{++}	Wen et al. [15]	2007				✓		✓				✓			✓
DecMiner	Lamma et al. [21,68]	2007						✓					✓		
AWS Mining	Greco et al. [26]	2008	✓				✓	✓			✓				✓
AGNEsMiner	Goedertier et al. [23]	2009	✓	✓	✓	✓	✓	✓		✓			✓		

β (or Tshingua α)	Wen et al. [16]	2009					✓	✓	✓		
Enhanced WFMiner	Folino et al. [35]	2009	✓	✓	✓	✓	✓	✓	✓		
EM-approach	Ferreira and Gilblad [36]	2009	✓							✓	
ILP Miner (Parikh)	van der Werf et al. [37]	2009			✓			✓	✓		✓
FSM Miner/Petrify	van der Aalst et al. [27,28]	2010		✓	✓	✓	✓	✓	✓		
FSM Miner/Genet	Carmona et al. [29]	2010					✓		✓		

cannot be qualified as robust, which is an important requirement in order to apply techniques in real-life settings.

2.2. *The* α-algorithm *and its successors*

The α-algorithm can be considered as one of the most substantial techniques in the process mining field. van der Aalst et al. [2] prove that it can learn an important class of workflow nets, called structured workflow nets, from complete event logs. The α-algorithm assumes event logs to be complete with respect to all allowable binary sequences and assumes that the event log does not contain any noise. Therefore, the α-algorithm is sensitive to noise and incompleteness of event logs. Moreover, the original α-algorithm was incapable of discovering short loops or non-local, non-free choice constructs. Alves de Medeiros et al. [14] improved the original α-algorithm to mine short loops and named it α^+. This algorithm is available in ProM and therefore, we included this algorithm in Table 1.

More techniques have been proposed to improve the original α-algorithm. Wen et al. [15] devised the α^{++}-algorithm that is able to detect non-free choice constructs. Furthermore, Wen et al. [16] have also proposed to take advantage of both start and completed event types in order to detect concurrency, which is implemented in their β-algorithm.

In order to remedy the robustness problem of the α-algorithms, Weijters et al. [17] developed HeuristicsMiner. In particular, HeuristicsMiner extends the formal α-algorithm in that it applies frequency information with regard to three types of relationships between activities in an event log: direct dependency, concurrency and not-directly-connectedness. According to its specification, HeuristicsMiner can discover short loops and non-local dependencies, but lacks the capability of detecting duplicate activities. As a result of the threshold parameter setting, the heuristic algorithm is prone to be noise resilient and therefore can be expected to be robust in a real-life context.

In contrast to the theoretically grounded α-algorithm, Günther and van der Aalst [18,68] propose Fuzzy Miner, an adaptive simplification and visualization technique based on significance and correlation measures to visualize the behavior in event logs at various levels of abstraction. The main contribution of Fuzzy Miner is that it can also be applied to less structured, or unstructured processes of which the event logs cannot easily be summarized in concise, structured process models. Although this approach is an extraordinary data exploration technique, it suffers from a drawback in the sense that a Fuzzy model cannot be translated to a formal Petri net which severely limits a comparative evaluation to other process discovery techniques.

2.3. *Techniques originating from machine learning theory*

Many authors have proposed to use machine learning techniques in the context of control-flow discovery. Several authors have used classification techniques for the purpose of process discovery. For instance, Maruster et al. [19] were among the first to investigate the use of rule-induction to predict dependency relationships between activities from a corpus of reference logs that portray various levels of noise and imbalance. To this end, the authors use a propositional rule induction technique, i.e. the uni-relational classification learner RIPPER, on a table of direct metrics for each process task in relation to each other process task, which is generated in a pre-processing step.

Ferreira and Ferreira [20] apply a combination of ILP learning and partial-order planning techniques to process mining. By iteratively combining planning and learning, a process model is discovered that is represented in terms of the case data preconditions and effects of its activities. In addition to this novel process mining technique, the contribution of this work is in the truly integrated BPM life cycle of process generation, execution, re-planning and learning. Lamma et al. [21,68] also describe the use of ILP to process mining. The authors assume the presence of negative sequences to guide the search algorithm. Unlike the approach of Ferreira and Ferreira, who use partial-order planning to present the user with an execution plan to accept or reject (a negative example), this approach does not provide an immediate answer to the origin of the negative events.

Due to the limitations of local search, early approaches to process discovery were generally not able to discover non-trivial constructs like non-free choice, invisible tasks and duplicate tasks. The main motivation of Alves de Medeiros et al. [22] to apply a genetic algorithm for process discovery is to benefit from global search. The fitness function of this genetic algorithm incorporates both a recall and a precision measure that drives the genetic algorithm towards suitable models. Genetic Miner defines its search space in terms of causal matrices. These matrices express task dependencies only, yet they are closely related to Petri nets. Because of the global search property, Genetic Miner is capable of detecting non-local patterns in the event log. Moreover, the algorithm ensures a fair robustness because of the arc post-pruning step.

AGNEsMiner [23], proposed by Goedertier et al., addresses the difficulties of process mining by representing the discovery task as first-order classification learning on event logs, supplemented with artificially generated negative events. Like Genetic Miner, the AGNEs algorithm is capable of constructing Petri net models from event logs and has been implemented as a mining plugin in the ProM Framework. Given an event log supplemented with artificially generated negative events, it becomes possible to learn the conditions that discriminate between the occurrence of either a positive or a negative event. As such, process mining is represented as a classification learning problem. The AGNEs process discovery algorithm is designed to learn the discriminating conditions that determine whether an event can take place or not, given a history of events of other activities. To induce this knowledge, AGNEsMiner makes use of an existing multi-relational classification learner called Tilde [24].

Another approach based on machine learning theory was proposed by Greco et al. [25]. This technique, called DWS mining, can be described as a hierarchical and iterative procedure that refines the process model in each step, based on clustering of patterns sharing similar behavior. This approach guarantees full compliance with the event log and increasingly improves the soundness of the process model, where soundness can be seen as a quantification of precision. In [26], Greco et al. extend traditional discovery methods by putting forward an abstraction method that aims to produce a taxonomy of process models. As such, the behavior in the event log can be analyzed at different levels of detail. AWS mining consists of both mining and abstraction algorithms that are assembled in order to build a tree-like schema. This schema consists of non-leaf nodes with an abstract process model that generalizes all the different process models in the corresponding subtree.

The idea to represent mined process models through different views at different levels of abstraction is also the driver behind FSM Miner/Petrify, a process discovery technique proposed by van der Aalst et al. [27,28]. In search of finding a balanced trade-off between precise and general process models, the authors propose a two-step approach. Firstly, a transition system should be constructed from the traces in an event log. In a second step, this transition system is synthesized by means of theory of regions. In this way, a Petri net can be constructed. Due to the more direct relation between the log and the transition system, generalization is more controllable. An important disadvantage of this approach is that it cannot deal with noise. A similar technique is described by Carmona et al. [29]. This algorithm, called Genet, also enables the construction of a Petri net from a transition system.

2.4. Other process discovery approaches

The last part of this section enumerates some other process discovery techniques that were not described previously.

Herbst and Karagiannis [30] describe the working of the splitpar algorithm that is part of the InWoLvE framework for process analysis. This algorithm derives a so-called stochastic activity graph and converts it into a structured process model. The splitpar algorithm is capable of detecting duplicate activities, but it is not able to discover non-local dependencies. Furthermore, Gaaloul et al. [32] propose a process discovery algorithm they called Workflow Miner. In fact, this technique enables the discovery of advanced structural workflow patterns after modeling the elementary dependencies in the event log in an intermediary graphical representation.

Furthermore, van Dongen and van der Aalst showed in [31] how process discovery can be approached by aggregating individual process instance models into a Petri net. For this approach, they rely on the use of Event-driven Process Chains (EPCs) as an intermediate step in order to derive an executable process model.

Another approach to process mining is the Enhanced WFMiner [35]. This algorithmic technique is described as being capable of dealing with a large number of important process discovery challenges such as noise, duplicate tasks and non-free choice. Moreover, Ferreira and Gillblad [36] address the problem of unlabeled event logs, which are logs without a case identifier for each process execution. By applying an Expectation–Maximization procedure, their technique allows to discover process models from an unlabeled stream of events.

Finally, ILP Miner [37] entails the application of Integer Linear Programming (ILP) to process discovery. This technique, formerly known as Parikh language-based region miner, is based on the well-known theory of regions and since this approach allows for parallelization and is shown to be independent of the number of events registered in the event log, the technique can be expected to be useful in practice.

3. Evaluating accuracy

Since the purpose of this paper is a multi-dimensional quality assessment of process discovery techniques, appropriate metrics need to be at hand in order to conduct this evaluation. In this section, we present an overview of how discovered process models can be judged with respect to their accuracy. However, accuracy in itself is multi-dimensional, which brings about a multitude of available metrics. Furthermore, in order to compare different process discovery techniques, it is indispensable to have some kind of framework to combine accuracy evaluation dimensions. In this section, it is argued that the F-score [38] can be employed in order to combine different types of accuracy metrics.

3.1. Accuracy dimensions

The accuracy of a process model is a somewhat abstruse concept because it can only be assessed taking into account different perspectives. As such, we identify three dimensions that are important: recall, precision, and generalization. It should be noted that process discovery techniques should be able to meet multiple criteria at the same time. For instance, techniques that only strive for maximizing recall tend to provide underfitting models that allow much more behavior than actually desired. Therefore, the key challenge for any process discovery technique is to discern process models that are able to find a balance between multiple dimensions in order to provide useful and understandable models.

Furthermore, the accuracy of discovered process models can be assessed using different methods [39]. Comparing the event log with the mined process model is one very popular approach that allows to devise metrics for various accuracy dimensions. In the literature, this type of evaluation is denoted as conformance checking [40]. More specifically, metrics that are based on the principle of comparing the behavior in the log with the behavior in the model can be referred to as model-log metrics. An overview of existing model-log metrics proposed can be found in Table 2. In this study, we will make use of this type of metrics in order to evaluate a selection of preeminent process discovery techniques.

Despite the fact that model-log metrics are a highly suitable quantification approach, there exist other ways for process model evaluation. An alternative for model-log metrics are metrics that quantify the difference between two process models [44]. This type of metrics can be used to compare an existing process model with a discovered model. However, it should be noted that these metrics cannot be employed for this evaluation study because we do not have any ex ante process models.

Finally, a couple of totally different evaluation approaches are available that have their origins in the field of data mining. Rozinat et al. show in [39] how Hidden Markov Models (HMMs) can be employed for process discovery evaluation. Furthermore, Calders et al. [45] propose to evaluate the quality of mined process models by making use of the minimum description length (MDL) principle. Although these are interesting approaches, we will not apply these methodologies in our study, which is mainly due to limited applicability in a real-life setting.

In the next paragraphs, different accuracy dimensions and corresponding metrics will be explained. It is important to notice that no single quantification of generalization has been proposed in the literature. This is of course due to the fact that process discovery is exploratory in nature owing to the absence of negative examples in the data. This makes the quantification of generalization a persistent challenge for process mining research. However, the availability of different precision metrics allows to investigate the way process models underfit the data. Since underfitting is typically a more common problem for process discovery techniques, the proposed methodology remains a defensible approach. Furthermore, we will propose a method to combine two evaluation dimensions in order to evaluate the overall accuracy of discovered process models.

Table 2
Overview of process discovery accuracy dimensions and metrics.

Dimension	Name	Symbol	Author	Available in ProM	Range	Model input type
Recall	Fitness	f	Rozinat and van der Aalst [40]	✓	[0,1]	Petri net
	Completeness	$PF_{complete}$	Alves de Medeiros et al. [34]	✓	$[-\infty,1]$	Heuristic net
	Completeness		Greco et al. [25]		[0,1]	Workflow schema
	Parsing measure	PM	Weijters et al. [17]	✓	[0,1]	Heuristic net
	Successfully executed traces	set	Rozinat [41]	✓	[0,1]	Petri net
	Behavioral recall	r_B^p	Goedertier et al. [23]		[0,1]	Petri net
Precision	Behavioral appropriateness	a_B	Rozinat and van der Aalst [40]	✓	[0,1]	Petri net
	Advanced behavioral appropriateness	a'_B	Rozinat and van der Aalst [40]	✓	[0,1]	Petri net
	Soundness		Greco et al. [25]		[0,1]	Workflow schema
	Behavioral specificity	s_B^n	Goedertier et al. [23]		[0,1]	Petri net
	Behavioral precision	p_B	De Weerdt et al. [42]		[0,1]	Petri net
	ETC precision	etc_P	Munoz-Gama and Carmona [43]	✓	[0,1]	Petri net

3.1.1. Recall

Recall or sensitivity is undoubtedly reckoned as the most important evaluation dimension of discovered process models. A recall metric reflects how much behavior present in the event log is captured by the model. For every process discovery algorithm, it is of utmost importance to render models with good recall because representing the control-flow behavior in an event log is the major objective of any discovery technique.

In recent years, a number of authors proposed metrics for quantifying the recall of a discovered process model with respect to the event log. A well-known recall metric is fitness f [40]. This Petri net based metric punishes for missing and remaining tokens when replaying the event log in the discovered process model. Although often used, there exist different alternatives for the fitness metric. For example, Weijters et al. [17] proposed the Parsing Measure *PM*, a much more coarse grained metric, that quantifies the percentage of traces that can be replayed in the discovered process model. Other valuable recall metrics are completeness ($PF_{complete}$) [22] and an alternative completeness metric as defined by Greco et al. [25].

Originating from their technique allowing to inject artificial negative events into an event log, Goedertier et al. [23] defined two evaluation metrics. One of these metrics is Behavioral Recall (r_B^p), which captures the percentage of correctly classified positive events in the event log by the discovered process model. This metric is more or less similar to fitness.

3.1.2. Precision

The key challenge for any process discovery technique is to come up with sensitive process models that at the same time find the right balance between underfitting (overly general process model) and overfitting (overly precise process model). The precision evaluation dimension gauges whether a mined process model does not underfit the behavior present in the event log. Some process models severely underfit the data by allowing a multitude of activity traces that are not appearing in the event log. Such models are typically not useful because they do not capture the restrictions of the process at hand very well.

In the literature, several precision metrics have been proposed. Greco et al. [25] defined soundness. Soundness is the percentage of traces compliant with the process model that have been registered in the log. Calculating soundness is not straightforward because enumerating all possible paths in a process model is hard. Even for smaller process models, it might be impossible to determine all the traces that are compliant with a process model.

A by far more used precision metric is the Advanced Behavioral Appropriateness (a'_B) as defined by Rozinat et al. [40]. Although this metric is theoretically sound in order to evaluate the precision of a process model, we have illustrated previously that this precision quantifier suffers from several drawbacks [46]. For instance, the calculation requires an exhaustive simulation which is computationally very demanding. Moreover, the implementation of this exhaustive simulation for calculating the metric within the ProM framework [7] is only approximate.

As mentioned before, Goedertier et al. [23] propose a technique that allows to inject artificial negative events into an event log. These artificial negative events can be used to define state-of-the-art process discovery evaluation metrics. By taking into account the classifications of the negative events only, a specificity metric, denoted as Behavioral Specificity (s_B^n), was introduced first. Very similar to this metric is Behavioral Precision (p_B) as defined in [42], which makes use of a combination of both misclassified positive and misclassified negative events.

Finally, Munoz-Gama and Carmona [43] recently proposed the *ETCPrecision* (etc_P) metric that takes into account the frequency of so-called escaping edges. This metric avoids a full state exploration and is implemented as a plug-in in ProM 6.

3.1.3. Combining precision and recall: F-score

A crucial problem of process model conformance checking is the absence of an evaluation framework. In [42], we proposed a first approach to combine evaluation metrics in order to evaluate the overall accuracy of a process model. This approach consists of applying the F-measure to both a recall and a precision metric.

Originally, the F-measure (Eq. (1)) was proposed by van Rijsbergen [38] in the context of information retrieval. In the fields of machine learning and data mining [3], the F-measure is often used as a standard balance between precision and recall for evaluating point classifiers

$$F_\beta = \frac{(1+\beta)^2 \times \text{precision} \times \text{recall}}{\beta^2 \times \text{precision} + \text{recall}} \tag{1}$$

The β-parameter is a weight factor to influence the relative importance of precision and recall. With β equal to 1, precision and recall are weighted evenly. This results in the F1-score. However, because process models discovered from real-life event logs are used, it might be interesting to change the weight factor in favor of recall. In Section 6, we will also report on the F2-score, weighing recall twice as much as precision. This is because especially for real-life event logs, recall tends to be more important than precision.

Furthermore, it should be noted that the application of the F-measure allows to combine a recall and a precision metric, but it does not take into account specificity nor structure. Nevertheless, recall and precision are definitely the most important accuracy evaluation dimensions. In order to counter this minor shortcoming of the F-measure, we will also report on specificity and structure in the results section.

4. Evaluating comprehensibility

As an evaluation criterion for mined process models, comprehensibility has received modest attention in the literature. Nonetheless, some articles have indicated that the available process discovery algorithms face a crucial problem with discovering comprehensible process models from highly unstructured event logs [47,48,28]. An important reason for the inferior interest in comprehensibility is the lack of good quantification methods for process model complexity. Moreover, the issue of comprehensibility is of inferior value when taking into account artificial event logs. Typically, process models mined from such event logs are smaller and much less complex than models mined from real-life event logs.

Within the domain of process modeling, Mendling et al. [49] have done much empirical work on the understandability of process models. They characterize understandability in terms of personal, structural and textual factors. They also found that experts with a more elevated familiarity with concurrency typically are better in understanding process models. Furthermore, an important insight of their study is the relation between structuredness and understandability. The most important result for our study is the major influence of model size on understandability.

4.1. *Comprehensibility metrics*

Although Mendling et al. describe interesting results with respect to comprehensibility, they do not provide metrics that quantify understandability. Therefore, we will employ metrics that were proposed by Lassen and van der Aalst. In [50], they describe three metrics for quantifying model complexity: Extended Cardoso Metric (ECaM), Extended Cyclomatic Metric (ECyM) and Structuredness Metric (SM). The first metric is based on a metric proposed by Cardoso [51] which takes into account certain splits and joins within the process model. The ECaM inherently perceives process model complexity in a local way because the metric only takes into account the successor nodes of each place. The ECyM builds further on work of McCabe [52]. In contrast to ECaM, the ECyM is based on the state space of the process model. This characteristic puts a strain on its applicability in a real-life environment because calculating the state space of complex process models often turns out to be infeasible. Finally, SM is a new metric that identifies behavioral patterns in a process model and scores these patterns according to their type and elements. The authors state that their SM is a better reflection of complexity because it takes into account penalties for components that are embedded in others. Lassen and van der Aalst implemented their metrics in a plugin for the ProM-framework. It should be pointed out that these metrics are designed for safe and sound WF-nets. The main reason is that the metrics are developed in the context of process modeling, rather than process mining. Because most process mining algorithms cannot guarantee to produce safe and sound WF-nets, we should take this limitation into account in the results section.

Next to ECaM, ECyM and SM, we will assess process model comprehensibility by also reporting more straightforward indications of model complexity in terms of Petri net building blocks such as number of transitions, number of places and number of arcs. Furthermore, the amount of a number of control-flow constructs are reported: the number of AND-Splits/Joins and the number of OR-Splits/Joins. The inclusion of these measurements is driven by the findings in [49,68]. It should be noted that these net characteristics are influenced by the type of algorithm. For example, the genetic algorithms and HeuristicsMiner produce Heuristic nets. When these nets are converted into Petri nets, a lot of invisible transitions are created, mainly for routing purposes. This creates a certain bias with respect to employing these metrics for comprehensibility evaluation.

5. Empirical setup

The first and major objective of this study is to assess process discovery techniques with respect to accuracy and comprehensibility. In order to do so, we will apply most of the metrics presented in Sections 3 and 4 to a subset of techniques depicted in Section 2. Furthermore, we will apply advanced statistical analysis techniques in order to compare the results and draw general conclusions. The empirical setup of the study is described in the following paragraphs.

5.1. *Process discovery techniques*

We assessed a total of seven state-of-the-art process discovery techniques: α^{+}, α^{++}, AGNEsMiner, Genetic Miner, Duplicate Task (DT) Genetic Miner, HeuristicsMiner and ILP Miner. Furthermore, we also report on the results of a flower model for each of the event logs, which is a process model that allows for any combination of activities and thus is a maximum sensitive but highly imprecise process model. It was infeasible to incorporate all process discovery techniques from Section 2 because of three important reasons. First of all, not every proposed algorithm is publicly available. Secondly, in order to evaluate process models, it is required that the technique presents results in the form of a Petri net or that the result is transformable to a Petri net. Finally, many techniques lack scalability and therefore do not allow to calculate models based on real-life event logs which are typically larger and more complex than artificially constructed event logs.

5.1.1. *Parameter settings*

Although many of the selected techniques require parameter tuning, we opted to employ standard settings as much as possible. However, for Genetic Miner and DT Genetic Miner, we reduced the population size to 10 and increased the maximum number of generations to 5.000 for the real-life event logs. This is according to the experiments in [22]. Furthermore, HeuristicsMiner was configured to also discover long distance dependencies. Finally, for AGNEsMiner, the injection probability π was lowered to 0.04 consistent with [47].

It should be noted that parameter tuning of some of these algorithms could potentially increase the performance. However, changing the parameters often requires prior knowledge that is seldomly available. Due to relatively high calculation times for both the process models as well as the evaluation metrics, especially for the more complex real-life event logs, a parameter tuning phase is practically infeasible. Moreover, for techniques like Genetic Miner and

AGNEsMiner, there are over 10 parameters. This makes the inclusion of parameter tuning even more infeasible.

5.2. *Artificial and real-life event logs*

Evaluation of process discovery algorithms traditionally relied on artificial event logs. For example, Alves de Medeiros et al. [22] and Goedertier et al. [23] report performance of different algorithms on artificial logs in order to demonstrate some specific requirements for process discovery algorithms such as dealing with parallel behavior, loops, non-local dependencies, etc. Alves de Medeiros et al. also present some results on four small scale real-life event logs. However, to our knowledge, this study is the first to provide a comprehensive benchmarking study of process discovery algorithms using real-life event logs.

In order to present a thorough assessment of process mining techniques, we will evaluate the algorithms both on artificial as well as on real-life event logs. Table 3 provides some details on 20 artificial event logs, which have been used previously by Alves de Medeiros et al. [34] to evaluate the Genetic Miner algorithm.

The real-life event logs are described in Table 4. These logs originate from information systems of different organizations: a university, a telecom company, a manufacturing company and an insurance company. All the event logs are recordings of human-centric processes and therefore exhibit a medium to high level of unstructuredness. In order to characterize the data sets, a number of event log properties are presented in Table 4, such as the

Table 3
Artificial event log properties.

Name	Activity types	≠ Process inst.	Process inst.	∥ Activity types	Loop	Skip	Non-free choice	Duplicate tasks
a10skip	12	6	300	1		✓		
a12	14	5	300	2				
a5	7	13	300	1	✓			
a6nfc	8	3	300	1			✓	
a7	9	14	300	4				
a8	10	4	300	1				
betaSimplified	13	4	300	0		✓	✓	✓
choice	12	16	300	0				
DriversLicense	9	2	300	0				
DriversLicensel	11	87	350	1	✓	✓	✓	✓
herbstFig3p4	12	32	300	3	✓			
herbstFig5p19	8	6	300	1				✓
herbstFig6p18	7	153	300	0	✓			
herbstFig6p31	9	4	300	0				✓
herbstFig6p36	12	2	300	0			✓	
herbstFig6p38	7	5	300	3				✓
herbstFig6p41	16	12	300	4				
l2l	6	10	300	0	✓			
l2lOptional	6	9	300	0	✓			
l2lSkip	6	8	300	0	✓			

Table 4
Description of the real-life event logs with following characteristics: the number of process instances (# PI), number of events (# EV), the number of activity types (# AT), the number of distinct process instances (# DPI), level of detail (LoD), structure (ST) and mean affinity (MA).

Label	Event log properties							Organization	Process description
	# PI	# EV	# AT	# DPI	LoD	ST	MA		
UFM	17.812	83.286	42	1.908	4.11	0.70	0.06	Telecom company	Second-line customer-initiated processes regarding the resolution of internet and telephony service problems
P2P	10.487	97.873	23	76	9.29	0.90	0.86	K.U.Leuven	SAP-based workflow process for handling invoices
KHD	1.541	8.657	18	251	5.38	0.74	0.24	K.U.Leuven	Status flow log of the ICT Service Helpdesk process
SIM	956	11.218	22	212	10.41	0.76	0.56	Manufacturing company	International Service Helpdesk processes regarding ICT-related problems
XBM	6.407	38.380	18	155	5.93	0.69	0.36	Insurance company	Interactive information sharing and resolution solving processes with remote brokers
XOA	2.004	12.432	49	71	6.00	0.95	0.11	Insurance company	Internal process of handling calls of tender
XNB	8.384	51.752	163	277	6.09	0.98	0.04	Insurance company	Internal back-office process of administering new commercial contracts with both private as well as business clients
XAO	614	3.631	14	36	5.90	0.79	0.38	Insurance company	Back-office handling of claims regarding occupational injuries

number of process instances (# PI), number of events (# EV), number of activity types (# AT) and the number of distinct process instances (# DPI). Furthermore, we also show three important metrics proposed by Günther [18]: level of detail (LoD), structure (ST) and mean affinity (MA). Level of detail is defined as the mean number of event classes per trace in the event log. Structure denotes the amount of observed behavior compared to the amount of theoretically possible behavior. A low value on structure indicates a more demanding challenge for process discovery techniques because it is very difficult to represent unstructured behavior in a sensitive and comprehensible process model. Mean affinity is somewhat similar to structure as this metric quantifies the mean relative overlap of direct following relations between each two traces in the event log. Again, low mean affinity values indicate an elevated difficulty for process discovery techniques. From Table 4, it can be seen that event logs UFM and XNB are especially problematic with respect to the diversity of behavior recorded.

5.3. *Statistical testing*

For analysis purposes, we make use of a collection of statistical techniques. These techniques provide mathematical ground for the conclusions.

5.3.1. *ANOVA, Friedman and Bonferonni–Dunn*

In order to compare the performance of multiple algorithms on manifold data sets, we make use of simple ANOVA (ANalysis Of VAriance) and a procedure described in Demšar [53]. In a first step of this procedure, the Friedman test [54] is performed which is a non-parametric equivalent of the well known ANOVA test. The null hypothesis of the Friedman test states that all techniques perform equivalent. The test statistic is defined as

$$\chi_F^2 = \frac{12P}{k(k+1)}\left[\sum_{j=1}^{k} R_j^2 - \frac{k(k+1)^2}{4}\right]$$

with R_j the average rank of algorithm $j = 1,2,\ldots,k$ over P data sets. Under the null hypothesis, the Friedman test statistic is distributed according to χ_F^2 with $k-1$ degrees of freedom, at least when P and k are big enough ($P>10$ and $k>5$), but in case of the real-life event logs, the exact critical values are used based on an adjusted Fisher z-distribution.

If the null hypothesis of equivalent performing techniques is rejected by the Friedman test, a post hoc Bonferroni–Dunn test [55] is applied to compare the process discovery techniques. The post hoc Bonferroni–Dunn test is a non-parametric alternative to the Tukey test and is defined as

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6P}}$$

with critical value q_α based on the Studentized range statistic divided by $\sqrt{2}$, and an additional Bonferroni correction by dividing the confidence level α by the number of comparisons made, $\alpha/(k-1)$, to control for family wise testing. This results in a lower confidence level and thus in higher power. The difference in performance between the best performing technique and other techniques is significant if the corresponding average ranks differ by at least the Critical Distance (CD).

5.3.2. *Principal component analysis and canonical correlation analysis*

In order to provide broader insights into the multi-dimensional quality assessment, the results of both a Principal Component Analysis (PCA) and a Canonical Correlation Analysis will be reported. PCA [56] is a statistical technique that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of uncorrelated variables called principal components. Because it is one of the most popular data exploration techniques, it fits very well within the research domain of process mining, which is exploratory in nature as well.

Next to PCA, we will also employ Canonical Correlation Analysis. This data analysis technique, introduced by Hotelling [57], allows to identify commonalities between two sets of variables. Given the different evaluation dimensions and the various metrics, we will be able to provide insight into the relations between these dimensions.

6. Results

The results section is subdivided into three parts. First, the accuracy of process discovery algorithms on artificial event logs is discussed. In a second part, we report on the assessment of the same techniques to real-life event logs with a key focus on accuracy and comprehensibility. Finally, a statistical multivariate analysis is presented in order to come up with more general findings.

6.1. *Artificial event logs*

The accuracy results for the artificial event logs are summarized in Tables 5 and 6. Table 5 represents the average results of eight algorithms on 20 artificial event logs for eight different metrics. Note that, in this and all following tables, the subsequent notation is used: the **best** average performance is underlined and denoted in bold face for each metric. A one-tailed paired *t*-test was used to test the significance of the performance differences in terms of absolute results. A non-parametric procedure combining the Friedman and Bonferonni–Dunn tests was used to test the significance of performance differences in terms of the average ranks, which are denoted between brackets. Performances that are **not significantly different at the 95% level** from the top-ranking performance are tabulated in bold face. Statistically *significant underperformances at the 99% level* are emphasized in italics. Performances significantly different at the 95% level but not at the 99% level are reported in normal font.

From Table 5, it can be concluded that overall, there is little difference between the selected process discovery algorithms. However, a remarkable result is that HeuristicsMiner presents a significant underperformance with respect to recall. Also, the different precision metrics present somewhat contradictory results. In previous work

Table 5
Average accuracy results for the artificial event logs.

Technique	Recall			Precision			
	f	r_B^p	PM	a_B'	s_B^n	p_B	etc_P
AGNEsMiner	**0.995**	**0.998**	**0.926**	**0.813**	**0.980**	**0.924**	**0.913**
α^+	*0.969*	0.952	**0.753**	**0.873**	**0.964**	0.862	**0.918**
α^{++}	0.984	0.972	**0.823**	**0.879**	**0.982**	**0.921**	**0.952**
DT Genetic Miner	0.996	**0.999**	**0.911**	**0.778**	**0.984**	**0.914**	**0.952**
Genetic Miner	**0.998**	0.984	**0.927**	*0.737*	**0.987**	**0.936**	**0.943**
HeuristicsMiner	*0.973*	*0.959*	**0.778**	**0.809**	0.982	**0.907**	**0.945**
ILP Miner	**1.000**	**0.991**	na	**0.786**	0.972	0.873	**0.902**
Flower	1.000	1.000	1.000	0.219	0.000	0.117	0.134

Table 6
Average F1- and F2-scores for the artificial event logs, with average ranks between brackets.

Technique	Approach A				Approach B			
	r_B^p	p_B	$F1_{r_B^p,p_B}$	$F2_{r_B^p,p_B}$	f	a_B'	$F1_{f,a_B'}$	$F2_{f,a_B'}$
AGNEsMiner	**0.998** (**3.6**)	**0.924** (**3.5**)	**0.953** (**3.4**)	**0.976** (**3.3**)	**0.995** (**3.9**)	**0.813** (**3.6**)	**0.873** (**3.6**)	**0.932** (**3.6**)
α^+	0.952 (**4.6**)	0.862 (**4.7**)	0.892 (**5.0**)	0.922 (**5.0**)	*0.969* (**4.7**)	**0.873** (**3.6**)	**0.904** (**3.8**)	**0.935** (**4.1**)
α^{++}	0.972 (**4.2**)	**0.921** (**3.7**)	**0.941** (**3.8**)	**0.957** (**3.8**)	0.984 (**4.1**)	**0.879** (**3.3**)	**0.897** (**3.6**)	**0.916** (**3.5**)
DT Genetic Miner	**0.999** (**3.6**)	**0.914** (**3.8**)	**0.949** (**3.8**)	**0.977** (**3.5**)	0.996 (**3.8**)	**0.778** (**4.1**)	**0.851** (**4.1**)	**0.922** (**4.0**)
Genetic Miner	0.984 (**3.8**)	**0.936** (**3.6**)	**0.954** (**3.6**)	**0.970** (**3.7**)	**0.998** (**3.8**)	*0.737* (**5.0**)	0.829 (**4.8**)	**0.915** (**4.6**)
HeuristicsMiner	*0.959* (**4.5**)	**0.907** (**4.3**)	0.927 (**4.3**)	**0.944** (**4.4**)	*0.973* (**4.5**)	**0.809** (**4.1**)	**0.864** (**4.0**)	**0.918** (**4.1**)
ILP Miner	**0.991** (**3.7**)	0.873 (**4.5**)	0.919 (**4.2**)	**0.958** (**4.4**)	**1.000** (**3.3**)	**0.786** (**4.4**)	**0.850** (**4.2**)	**0.916** (**4.1**)
Flower	1.000	0.117	0.208	0.392	1.000	0.219	0.349	0.556

Table 7
Average accuracy results for the real-life event logs.

Technique	Recall			Precision			Run time
	f	r_B^p	set	a_B'	s_B^n	p_B	
AGNEsMiner	*0.744*	*0.861*	*0.362*	0.590	0.897	*0.363*	10 h:29 m:2 s
α^+	0.648	*0.551*	*0.033*	*0.491*	0.625	0.237	1 s
α^{++}	*0.523*	*0.373*	*0.015*	*0.433*	0.624	0.249	1 s
DT Genetic Miner	*0.751*	**0.871**	*0.383*	**0.697**	*0.896*	*0.329*	1 d:6 h:57 m:47 s
Genetic Miner	*0.524*	0.819	*0.227*	**0.713**	**0.935**	**0.429**	3 d:12 h:41 m:53 s
HeuristicsMiner	**0.775**	*0.729*	0.469	**0.778**	**0.960**	**0.516**	5 s
ILP Miner	**0.965**	**0.951**	**1.000**	0.593	0.766	*0.249*	2 m:8 s
Flower	1.000	1.000	1.000	0.352	0.009	0.049	1 s

[46], we have indicated some issues with a_B' which are due to the state space exploration that is required to calculate this metric. This study again indicates significant differences between a_B' and other precision metrics.

Table 6 presents the application of the F-measure to r_B^p, p_B (approach A) and f, a_B' (approach B) respectively. By examining both evaluation approaches, we can conclude that AGNEsMiner, DT Genetic Miner and Genetic Miner perform outstandingly, whereas the results of HeuristicsMiner, ILP Miner and the α-algorithms tend to indicate slight accuracy problems, given the setting of evaluation based on artificial event logs. HeuristicsMiner and the α-algorithms show an underperformance in recall, while ILP Miner accounts for less precise process models.

6.2. *Real-life event logs*

In contrast to the artificial event logs, the algorithms are assessed along two dimensions, namely accuracy and comprehensibility.

6.2.1. *Accuracy*

The accuracy results for the real-life event logs are displayed in Tables 7 and 8. The former presents the aggregate accuracy results. Note that in this table, the Parsing Measure (*PM*) is replaced by the number of successfully executed traces (abbreviated as *set*). This metric is a Petri net alternative for the Parsing Measure, which could only be calculated for about 50% of the real-life event log based

Table 8
Average F1- and F2-scores for the real-life event logs, with average ranks between brackets.

Technique	Approach A				Approach B			
	r_B^p	p_B	$F1_{r_B^p,p_B}$	$F2_{r_B^p,p_B}$	f	a_B'	$F1_{f,a_B'}$	$F2_{f,a_B'}$
AGNEsMiner	*0.861* (**3.4**)	*0.363* (**3.6**)	0.476 (**3.1**)	**0.612** (**2.8**)	*0.744* (**4.0**)	0.590 (5.1)	**0.642** (**4.3**)	**0.694** (**3.9**)
α^+	*0.551* (*5.4*)	0.237 (*5.7*)	*0.265* (*6.3*)	*0.338* (*6.6*)	0.648 (4.7)	*0.491* (4.8)	**0.547** (**4.9**)	*0.598* (**4.9**)
α^{++}	*0.373* (*6.6*)	0.249 (*5.3*)	*0.262* (*5.6*)	*0.296* (*6.3*)	*0.523* (*5.2*)	0.433 (*5.9*)	0.473 (**4.0**)	*0.501* (**5.1**)
DT Genetic Miner	**0.871** (**2.8**)	*0.329* (**4.0**)	0.457 (**3.9**)	**0.615** (**3.0**)	*0.751* (**3.9**)	**0.697** (**3.3**)	**0.703** (**4.6**)	**0.725** (**3.6**)
Genetic Miner	0.819 (**3.1**)	**0.429** (**3.0**)	**0.535** (**2.6**)	**0.652** (**2.5**)	*0.524* (*5.9*)	**0.713** (**3.0**)	*0.563* (**4.6**)	0.534 (5.5)
HeuristicsMiner	0.729 (5.0)	**0.516** (**1.9**)	**0.587** (**2.0**)	**0.656** (**2.6**)	**0.775** (**2.8**)	**0.778** (**1.8**)	**0.737** (**2.6**)	**0.753** (**2.5**)
ILP Miner	**0.951** (**1.8**)	*0.249* (**4.5**)	*0.371* (**4.5**)	*0.553* (**4.3**)	**0.965** (**1.6**)	*0.593* (**4.1**)	**0.699** (**3.0**)	**0.809** (**2.5**)
Flower	1.000	0.049	0.092	0.197	1.000	0.352	0.477	0.627

process models because the calculation of the metric requires a translation from a Petri net to a heuristic net which often failed. Furthermore, the etc_P-metric is not represented for the same reason since it could not be calculated for a large enough number of process models. Finally, we also added the average run time of each algorithm. Note also that the detailed results of the selected discovery techniques can be found in Appendix A.

From Table 7, it can be concluded that there is much more significant difference amongst the process discovery techniques. ILP Miner clearly outperforms the other algorithms in terms of recall, whereas HeuristicsMiner has the upper hand with respect to precision. From these results, it becomes lucid why the use of the F-score as an accuracy evaluation method becomes feasible. It provides a valuable approach to balance between recall and precision. By combining recall and precision values into one metric, the overall accuracy of discovered process models can be assessed.

The F-score results for the real-life event logs is shown in Table 8. From this table, it can be seen that HeuristicsMiner is the most powerful technique in a real-life environment. It outperforms the other algorithms for both evaluation approaches and according to both the parametric and non-parametric statistical evaluations. Only according to the $F2_{f,a_B'}$ metric where recall is weighed twice as much as precision, ILP Miner shows better performance. However, this is mainly due to the fact that the precision metric (a_B') overestimates the actual precision of the model. For a detailed analysis on the drawbacks of a_B', we refer to [46]. Another conclusion that can be formulated from Table 8 is that the α-algorithms are least suitable. This is not surprising because these algorithms are more theoretical in nature and therefore less robust in a real-life setting.

When comparing the performances on real-life event logs with those of the artificial logs, it becomes clear that there are significant differences between both. As for the artificial event logs, especially AGNEsMiner and the genetic algorithms rank at the top. However, for the real-life event logs, this picture shifts thoroughly as the HeuristicsMiner algorithm clearly outperforms these algorithms. What is also important to notice is the significant difference in average run time, which again favors the HeuristicsMiner algorithm.

6.2.2. *Comprehensibility*

Next to accuracy, a second focus of this evaluation study is mined process model comprehensibility. The metrics identified for evaluating comprehensibility are described in Section 4.1. Despite the choice for quantifying comprehensibility by using process model complexity metrics, it goes without saying that assessing model comprehensibility in a quantitative way is challenging. Because comprehensibility remains largely subjective, relying on a visual analysis is definitely worthwhile in practice.

In order to clarify the discussion on comprehensibility to some extent, Figs. 2 and 3 show a selection of the discovered process models. Fig. 2 shows the discovery results of a less complex real-life event log (XAO). From this figure, it can be seen that even for relatively simple event logs, the comprehensibility of the models can differ significantly. For instance, the ILP Miner result shows a significantly higher number of arcs with respect to the number of places and transitions. What is more, the transformation of the models discovered by HeuristicsMiner and Genetic Miner (i.e. heuristic nets) to Petri nets entails the inclusion of different invisible transitions for routing purposes, which obviously decreases comprehensibility.

In contrast to the relatively understandable process models in Fig. 2, the picture shifts when taking into account Fig. 3. The fragment of the ILP Miner model clearly shows a severe degree of incomprehensibility. The HeuristicsMiner result might be judged more comprehensible to some extent, but overall comprehensibility is moderate as well. This indicates that for highly complex event logs, even better performing techniques such as HeuristicsMiner and AGNEsMiner fail to address the challenge of complexity. From the visual analysis, it can be concluded that foremost the number of arcs with respect to the number of nodes is an appropriate indicator of model comprehensibility.

As indicated earlier, a quantitative approach for assessing comprehensibility was opted for. Table 9 provides the average results over the eight real-life event logs for each of the algorithms. For those metrics that are not in the [0,1]-interval, we scaled the results for each event log and afterwards averaged these scaled values. Notice that in this table, lower values signify better comprehensibility. Furthermore, the average ranks are again reported between brackets. Note also that the ECyM could not be reported as an average because the metric resulted in a high amount of infinite values. The main reason hereto is the infeasibility to calculate the state space of a certain mined process model. Therefore, we did not include this

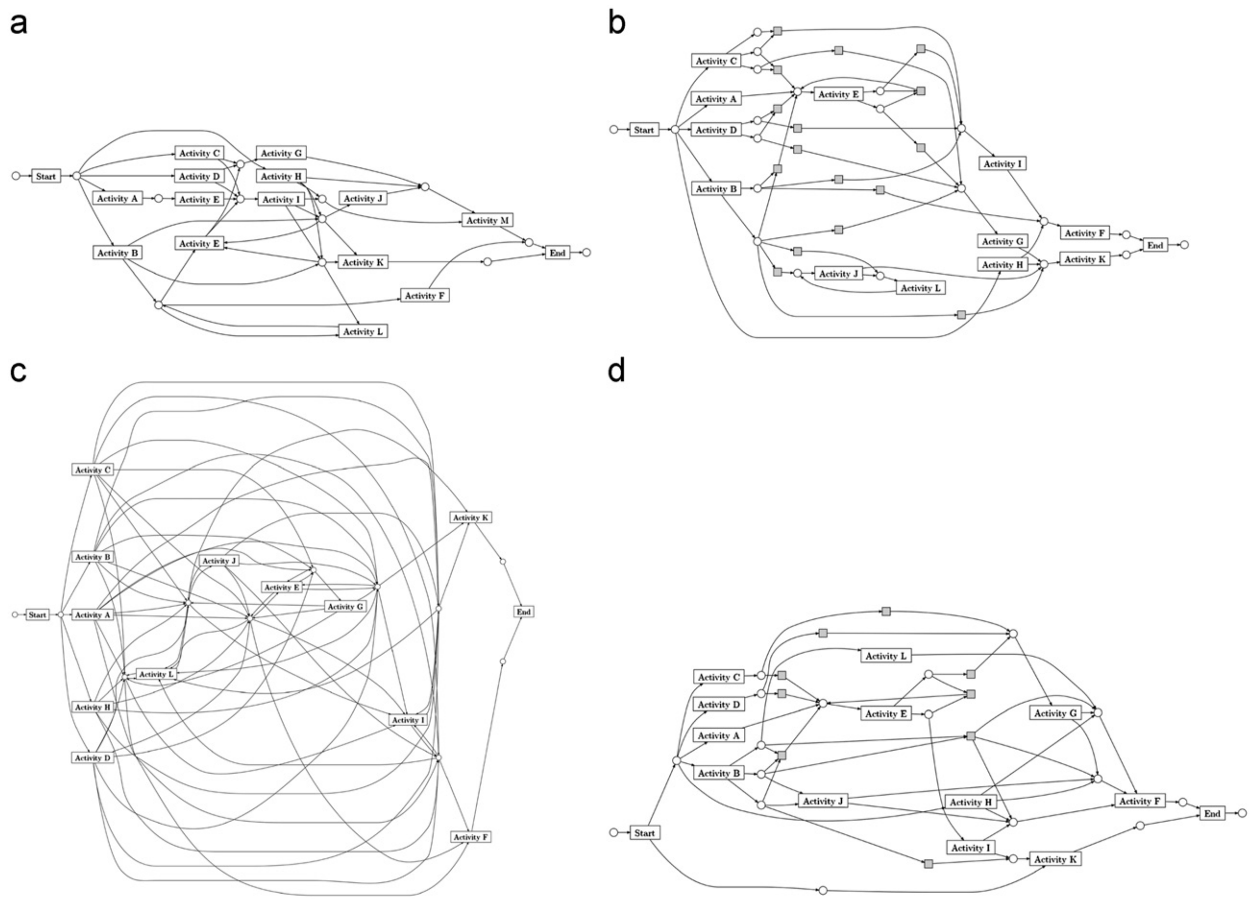

Fig. 2. Four discovered process models by different discovery algorithms for the XAO data set. (a) AGNEsMiner. (b) Genetic Miner. (c) ILP Miner. (d) HeuristicsMiner.

metric in the evaluation table. Note that the log-specific comprehensibility results can be found in Appendix B.

The comprehensibility results do not allow for a straightforward interpretation. Therefore, we will discuss a number of key observations. First of all, Genetic Miner is critically underperforming since all possible incomprehensibility indicators show that discovered process models by this algorithm are significantly more difficult to interpret than those of other process discovery algorithms. This adds to the fact that the algorithm is also problematic in terms of run time. Although Bratosin et al. [58] showed how the performance of genetic miner could be improved by applying a sampling procedure, we argue that the use of Genetic Miner in a real-life setting is inadvisable.

Secondly, regarding the Petri net building blocks, it can be seen that ILP Miner comes up with process models with an extremely high amount of arcs. More precisely, this algorithm creates on average 1633 arcs between transitions and places and vice versa. Whatever other metrics might indicate, this high amount of arcs renders any process model incomprehensible. As illustrated in Fig. 2, even for easier data sets such as XAO, the overfitting behavior of ILP Miner clearly complicates comprehensibility.

Another important remark is the major contradiction between ECaM and SM. Although both metrics are not able to indicate comprehensibility differences between the techniques when taking into account the average ranks, the absolute values of these metrics dissent severely. This difference might be due to the discrepancy in local or global analysis of complexity. However, we think that it also indicates that it is not very straightforward to devise comprehensibility metrics that are suitable for process models mined from real-life event logs.

Notwithstanding the visual inspection of the discovered process models, the results in Table 9 illustrate that it is challenging to prove significant differences between process discovery algorithms statistically. In addition, all of the presented metrics suffer from some kind of bias. Simple metrics such as number of places, arcs, AND-Joins/Splits and OR-Joins/Splits are influenced by the modeling language whereas the more advanced complexity metrics suffer from the difficulty to deal with highly unstructured process models which are less common within process modeling, the domain they were devised for. We can expect that by taking into account a multitude of metrics, these biases are slightly filtered out. Therefore, we think that this analysis remains valuable.

To conclude, it should be noted that in general, the comprehensibility of discovered process models is mediocre. Especially for the more difficult event logs like UFM

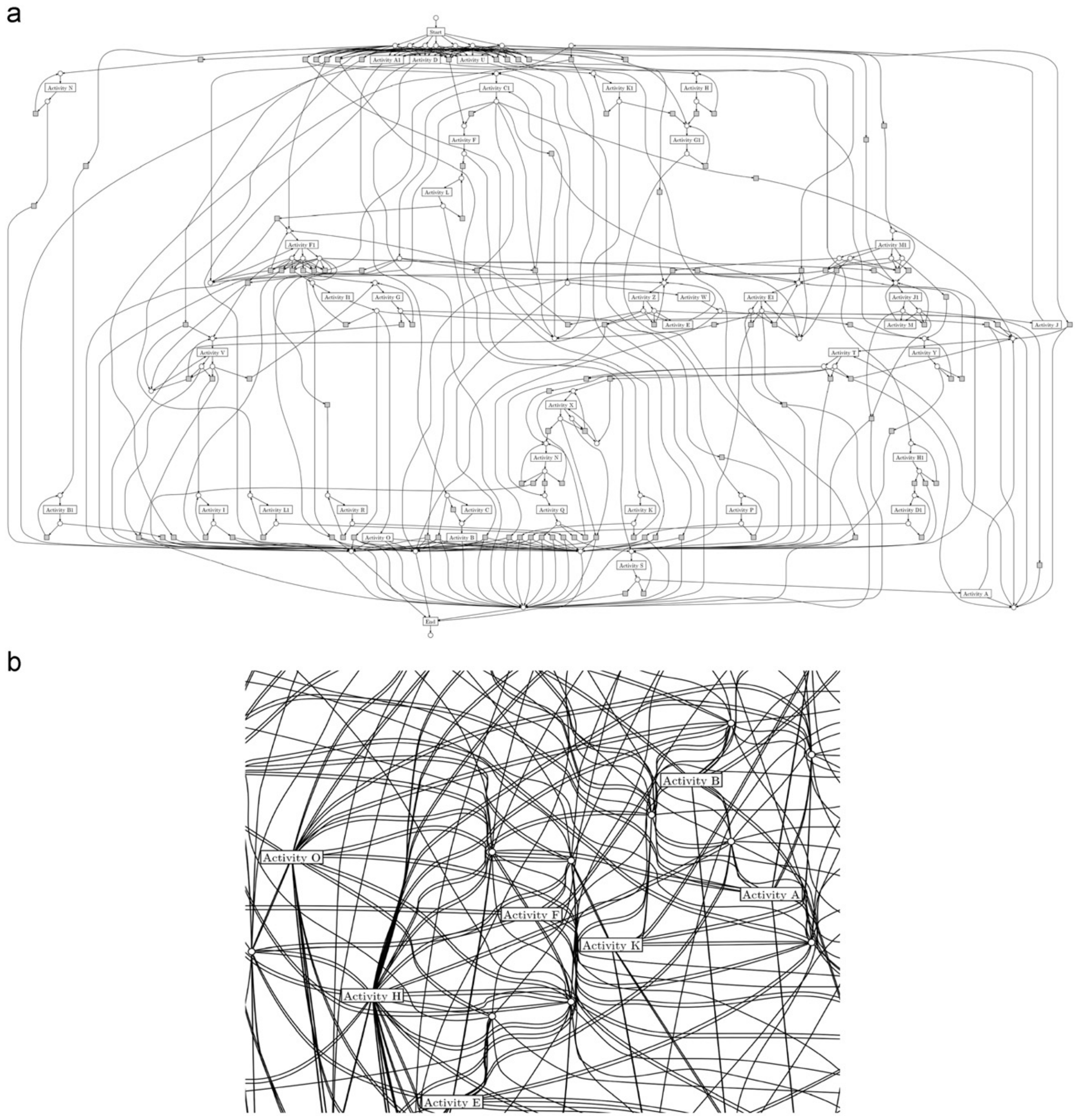

Fig. 3. Discovered process models for the complex UFM event log. (a) Petri net discovered by HeuristicsMiner. (b) Fragment of the result of ILP Miner, clearly showing overspecification.

and XNB, the discovered process models are highly unstructured and therefore uninterpretable, which shows that knowledge discovery from such data sets causes major problems for traditional process discovery techniques.

6.3. *Multivariate analysis*

In this final part of the results section, we provide the outcome of the application of two statistical analysis techniques, as described in Section 5.3. These techniques allow for the formulation of some more general findings regarding the evaluation of process discovery techniques in a real-life setting.

6.3.1. *Principal component analysis*

Principal component analysis is a variable reduction technique that is well suited to analyze a number of interrelated variables. In this case, we applied PCA on a data set consisting of 61 observations, each observation being the results of one process discovery algorithm on one data set. Note that for three algorithm event log combinations, the discovery algorithm could not learn a process model from

Table 9
Comprehensibility results for the real-life event logs.

Technique	Transitions	Places	Arcs	AND-Joins	AND-Splits	OR-Joins	OR-Splits	ECaM	SM
AGNEsMiner	*0.515* (**3.9**)	**0.332** (**2.4**)	**0.301** (**2.1**)	**0.402** (2.9)	**0.363** (**2.6**)	**0.267** (**2.6**)	**0.279** (**2.1**)	**0.275** (**2.5**)	**0.173** (**4.3**)
α^{+}	**0.354** (**2.6**)	**0.459** (**3.9**)	**0.367** (**3.3**)	**0.510** (**4.1**)	**0.515** (**4.2**)	**0.414** (**4.0**)	**0.374** (**3.8**)	**0.331** (**3.0**)	**0.257** (**3.2**)
α^{++}	**0.331** (**3.2**)	0.605 (**5.1**)	**0.641** (5.3)	**0.543** (**5.2**)	**0.633** (5.6)	0.757 (*6.1*)	0.710 (*5.6*)	**0.611** (**4.8**)	**0.240** (**4.8**)
DT Genetic Miner	*0.772* (5.3)	0.594 (**4.1**)	**0.400** (**3.0**)	**0.364** (**2.3**)	**0.392** (**2.6**)	**0.310** (**2.7**)	**0.309** (**2.6**)	**0.468** (**3.9**)	**0.245** (**3.6**)
Genetic Miner	*0.919* (*6.1*)	*0.834* (*6.1*)	*0.544* (**4.8**)	0.755 (**4.8**)	0.618 (**3.6**)	0.567 (**4.6**)	*0.656* (**4.9**)	*0.565* (**4.8**)	*0.585* (**5.3**)
HeuristicsMiner	*0.715* (5.2)	*0.482* (**3.9**)	**0.360** (**3.4**)	**0.633** (**4.4**)	*0.618* (**4.3**)	**0.364** (**3.0**)	0.410 (**3.4**)	0.372 (**3.8**)	0.209 (**4.5**)
ILP Miner	**0.381** (**1.9**)	**0.365** (**2.6**)	*0.866* (*6.1*)	**0.626** (**4.4**)	*0.742* (**5.1**)	*0.653* (**5.1**)	*0.696* (*5.7*)	*0.770* (**5.3**)	**0.140** (**2.4**)
Flower	0.409	0.046	0.141	0.000	0.000	0.031	0.036	0.021	0.002

Table 10
Loadings of the principal components.

Metric	**PC1**	**PC2**	**PC3**	**PC4**
$\mathbf{r}_B^p$	0.310	**0.695**	0.368	−0.168
f	−**0.515**	**0.707**	0.148	0.225
set	−0.206	**0.863**	0.169	0.112
$\mathbf{p_B}$	−0.077	−0.264	**0.762**	0.238
$\mathbf{a'_B}$	0.036	−0.494	**0.526**	0.403
$\mathbf{a'_S}$	0.112	0.070	−0.496	**0.808**
SM	**0.861**	−0.045	0.149	0.038
ECaM	**0.797**	0.404	−0.025	0.039
Transitions	**0.816**	−0.078	−0.078	−0.196
Places	**0.961**	−0.019	0.114	0.006
Arcs	**0.832**	0.389	−0.027	0.032
AND-Joins	**0.956**	0.036	0.043	−0.062
AND-Splits	**0.959**	−0.011	0.036	−0.026
OR-Joins	**0.956**	0.117	0.079	0.074
OR-Splits	**0.944**	0.166	0.041	0.081

the data. Our main goal is to find out whether a high number of both accuracy and comprehensibility metrics can be reduced to a manageable few without much loss of information. The analysis took into account 15 different metrics, as represented in Table 10.

Fig. 4 is the scree plot, as proposed by Cattell [59], which allows to identify the number of principal components (PCs) to retain. The figure shows an elbow point at PC3. However, because the eigenvalue of the fourth principal component also exceeds 1, we report the principal component scores for the first four PCs. These four principal components account for 83% of the variance of the original data.

Table 10 shows the loadings of the different principal components with respect to the original variables. Firstly, it can be seen that the first principal component (PC1) is highly correlated with all the comprehensibility variables. This PC can be seen as an aggregate of process model comprehensibility, which shows that the selected comprehensibility metrics correlate significantly. As such, by investigating the average principal component scores of PC1 for the different algorithms, we can conclude that Genetic Miner (2,39) and ILP Miner (0,58) can be considered more incomprehensible models whereas AGNEsMiner (−0,60) and HeuristicsMiner (−0,30) are better performing in this dimension. Table 10 further indicates that PC2 captures the variance in recall, whereas PC3 represents the precision of discovered process models. Finally, PC4 is strongly correlated with a'_S.

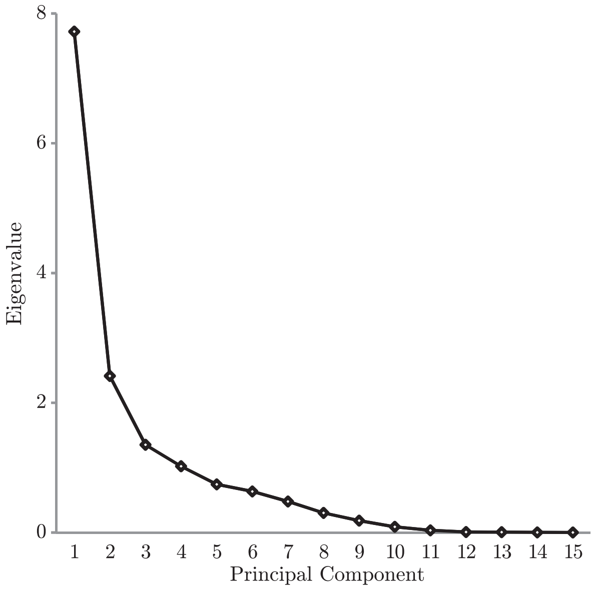

Fig. 4. Scree plot of the PCA showing an elbow point at PC3.

In conclusion, this analysis implies that it can be useful to combine different metrics into a single variable in order to evaluate a process model in respect to a certain dimension. However, this combination of metrics is strictly different from the F-measure approach introduced in Section 1, which is a method to find a balance between different accuracy dimensions.

6.3.2. *Canonical correlations*

Canonical correlation analysis is an appropriate statistical technique for analysis of relationships between two sets of variables. The technique, developed by Hotelling [57], accounts for the covariation of the variables from both within and across sets. Because we can expect to find correlations between accuracy, comprehensibility and log complexity, canonical correlation is a suitable method for analyzing the relationships between the dimensions proposed. The data set used consisted of the same 61 observations from the previous section extended with the event log characteristics described in Table 4.

The results of three separate canonical correlation analyses are presented, as shown in Table 11. These analyses differ from each other in terms of the choice of criterion and predictor variables. For every analysis, only the first canonical

Table 11
Criterion and predictor variables for the three different canonical correlation analyses, with the value and the significance of the first canonical correlation.

Analysis	Criterion variables	Predictor variables	1st Can. Corr.
CCA 1	Accuracy f, r_B^p, set, a'_B, p_B	Log complexity # PI, # EV, # AT, # DPI, LoD, ST, MA	0.79 ($F=2.26$ with $p=0.0002$)
CCA 2	Comprehensibility Places, arcs, AND-Joins, AND-Splits, OR-Joins, OR-Splits, ECaM, SM	Log complexity # PI, # EV, # AT, # DPI, LoD, ST, MA	0.75 ($F=1.63$ with $p=0.0068$)
CCA 3	Accuracy f, r_B^p, set, a'_B, p_B	Comprehensibility Transitions, places, arcs, AND-Joins, AND-Splits, OR-Joins, OR-Splits, ECaM, SM	0.76 ($F=1.86$ with $p=0.0021$)

variate pair was significant at the 0.01 level, as determined by the F-value of the Wilks' Λ-tests. This test is equivalent to the likelihood ratio test and it verifies the null hypothesis that the canonical correlations in the current row and all that follow are zero. From this, we can conclude that the analyses allude to the presence of interdependence between the different sets of variables.

It should be noted that large canonical correlations may not imply strong correlation between the criterion and predictor variables. In order to discuss the practical significance of the canonical correlation analyses, we make use of a redundancy measure as proposed by Stewart and Love [60]. This measure determines how much of the variance in one set of variables is accounted for by the other set of variables. As such, the proportion of variance in the accuracy variables predictable from the log complexity variables was 21%. Similarly, the proportion of variance in the comprehensibility variables predictable from the log complexity variables amounted up to 24%. Finally, about 28% of the variance in the accuracy variables is shared by the comprehensibility variables. Taking into account that a lot of variance in the criterion variables is accounted for by the type of algorithm applied, we can conclude that there is practical evidence that log complexity has an important impact on both accuracy and comprehensibility. Furthermore, a noteworthy correlation exists between accuracy and comprehensibility.

To conclude this canonical correlation analysis, we provide the correlations between the first canonical variable and the original variables in Table 12. These correlations, also known as *structural correlations*, allow for interpreting the canonical variates. For example, from the first canonical correlation analysis (CCA 1), we can see that the first canonical variate represents foremost Behavioral Precision (p_B) and to a lesser extent fitness (f). Similarly, for the log complexity predictor variables, it can be seen that both the number of distinct process instances (# DPI) and the number of process instances (# PI) have an important influence on the first canonical variate LogComplexity1. Thus, the shared variance between accuracy and log complexity is mainly expressed by the influence of # DPI and # PI on p_B and f.

Furthermore, for CCA 2 it can be seen that the first canonical variate Comprehensibility1 is strongly correlated with ECaM, number of arcs and number of AND-Joins/Splits and LogComplexity1 is predominantly correlated with # AT.

Table 12
Correlations between the first canonical variate and their original variables for each of the canonical correlation analyses (CCAs).

CCA 1 Accuracy 1		CCA 2 Comprehensibility 1		CCA 3 Accuracy 1	
f	−0.53	Places	0.38	f	−0.88
r_B^p	−0.41	Arcs	0.66	r_B^p	−0.63
set	−0.31	AND-Joins	0.64	set	−0.45
a'_B	0.21	AND-Splits	0.64	a'_B	0.27
p_B	−0.74	OR-Joins	0.38	p_B	−0.09
		OR-Splits	0.47		
		ECaM	0.68		
		SM	0.20		
LogComplexity 1		LogComplexity 1		Comprehensibility 1	
# PI	0.55	# PI	0.44	transitions	0.73
# EV	0.30	# EV	0.29	places	0.69
# AT	0.21	# AT	0.94	arcs	0.36
# DPI	0.75	# DPI	0.42	AND-Joins	0.75
LoD	−0.10	LoD	−0.29	AND-Splits	0.72
ST	−0.44	ST	0.29	OR-Joins	0.59
MA	−0.35	MA	−0.55	OR-Splits	0.58
				ECaM	0.28
				SM	0.57

These positive correlations bring about that the number of activity types has a notable influence on process model comprehensibility.

Finally, ascertaining the relationship between accuracy and comprehensibility in CCA 3, it can be concluded that there exists a strong negative relation between predominantly the recall variables (f, r_B^p) and the number of transitions, the number of places and the number of AND-Joins/Splits. Accordingly, the shared variance between comprehensibility and accuracy is primarily channeled through the recall variables.

7. Discussion

In Section 5, the quality assessment of process discovery techniques is restricted to algorithms that discover a Petri net or algorithms of which the resulting models can be transformed into a Petri net. There exist two important reasons why a Petri net based evaluation methodology is opted for. Firstly, a majority of state-of-the-art discovery algorithms comply with this requirement. Furthermore, from an evaluation perspective, the majority of metrics for assessing discovered process models is defined in terms of Petri nets. In this way, a Petri net based quality assessment is considered the most feasible approach to benchmarking process discovery algorithms.

7.1. Representational bias

It is important to note that the choice for Petri nets as the representational language for discovered process models might collide with this study's main goal to assess process discovery techniques based on real-life event logs. This is because especially for highly complex event logs exhibiting a large variety in process behavior, relying on Petri nets as the representational language might be less suited. It has been found (e.g. [47]) that Petri net based discovery techniques sometimes lack the capabilities to learn accurate and comprehensible models from this type of event logs. In such cases, alternative discovery techniques such as the Fuzzy Miner [18], or event log manipulation techniques (e.g. pattern abstractions [61,69], sequence clustering [25,62,63,68], etc.) are considered more useful because these techniques feature better abstraction capabilities. Despite the potential of alternative discovery techniques with increased abstraction capabilities going beyond the discovery of a single process model, they require a different evaluation methodology since abstraction itself needs to be taken into account as an extra evaluation dimension. It should be noted that research with respect to holistically benchmarking different approaches for the analysis of highly complex event logs is to the best of our knowledge nonexistent in the process mining literature. The issue of representational bias is of course strongly related to the chosen evaluation methodology. Accordingly, the main limitations of our approach are discussed in the next section.

7.2. Limitations of a Petri net based evaluation methodology

The limitations of the chosen representational bias, i.e. Petri nets, are twofold. Firstly, the representational bias might cause the inability to represent the underlying process model of a certain event log well. This is related to what is explained in the previous paragraph. Techniques that discover a single Petri net from an event log might be confronted with the problem that not even a single element in the search space is a satisfactory discovery result. As far as we know, no research has explicitly focussed on the ability of different representational biases to discover process models from event logs, let alone on the improvement of the representational bias for process discovery. Notwithstanding that design-oriented languages such as Petri nets or EPCs are often used for the representation of discovered process models, the modeling origin often complicates the process discovery learning problem. As such, it should be investigated how to increase the flexibility of the representational bias, for instance by using less restrictive languages such as fuzzy nets [18], declarative models [64] or C-nets [65].

A second limitation following the choice for Petri nets is that discovered models can be internally inconsistent (i.e. not sound). This is because none of the discovery techniques limit the search space to sound workflow nets (WF-nets [66]). As such, discovered process models might contain livelocks and/or deadlocks. The problem of soundness is strongly confirmed by this study as well, since none of the discovery techniques was able to discover a sound Petri net, even from the more straightforward real-life event logs. Notwithstanding the fact that unsoundness has a definite impact on both accuracy and comprehensibility of process discovery techniques, it remains debatable whether a Petri net based process discovery technique's first priority should be to render sound process models. Soundness is foremost important in the process modeling domain, whereas the main goal of process discovery techniques is to provide insight into a set of process executions. Since soundness is not the top priority problem in process discovery, we find that currently available process discovery techniques apparently do not consider soundness in their learning strategies. It can be argued that the importance of soundness is such that it should be incorporated in the process discovery learning problem. However, it is shown that the process discovery learning problem is highly challenging even without considering soundness. Accordingly, it can be advocated that due to low-level nature and additional criteria such as soundness, Petri nets might not be the most suitable representational bias for process discovery.

Despite the limitations of Petri net based discovery techniques, it is beyond the scope of this study to explore or assess the whole aspect of the representational bias in process discovery. Furthermore, it is rarely known upfront whether an event log will create major challenges for traditional process discovery techniques. As such, it remains very useful to know which discovery techniques perform better on real-life event logs, even if after initial analysis improved abstraction techniques may provide more useful results. In addition, many solutions for the complexity problem arising from event logs displaying a high variety of process behavior will inevitably make use of some sort of process notation. Despite the shortcomings, Petri nets remain a valid option for this task.

8. Conclusion

Assessing the quality of process discovery techniques is an essential element for both process mining research as well as for the use of process mining in practice. Up till now, evaluation of process discovery techniques was performed in very diverse ways, but mostly on artificial data sets. In this study, a multi-dimensional evaluation study was carried out in order to evaluate process mining techniques comprehensively on eight real-life event logs. The results of our study allow for the formulation of a number of key conclusions:

- There exists an important difference between evaluation of process discovery algorithms based on either artificial or real-life event logs. The use of real-life event logs for process discovery quality assessment is a major contribution of this study.

- The F-score methodology is a valuable approach to combine recall and precision metrics in order to assess overall accuracy of a process model.
- HeuristicsMiner seems the most appropriate and robust technique in a real-life context in terms of accuracy, comprehensibility, and scalability.
- It was statistically validated that dealing with high complexity event logs remains an important challenge for process mining algorithms, both in terms of accuracy as well as in terms of comprehensibility.
- Proficient quantification of discovered process model comprehensibility still faces fundamental challenges. Metrics from the process modeling domain were applied in this study, but were found to be difficult to interpret because of the low-level of structure many mined, real-life process models exhibit.
- Multivariate statistical analyses showed that it might be useful to reduce different metrics into one single quantification of a certain dimension. Furthermore, it was demonstrated that there exists important correlations between process model accuracy, process model comprehensibility and event log complexity.
- Representational bias is a widely overlooked research topic in the process mining domain. This study centering on single model Petri net discovery techniques shows that for complex real-life event logs, alternative analysis techniques can be expected to be an interesting follow-up analysis.

Overall, we think that the future of process mining research should emphasize on developing insightful techniques for analyzing real-life event logs. Although currently available control-flow discovery techniques can still play a significant role in conducting process mining analysis in practice, more complex event logs require novel data exploration techniques with a definite focus on comprehensibility. For instance, Bose and van der Aalst [67] show how trace alignment might be used to derive insights from complex event logs. In future work, we aim at developing original data exploration techniques that allow for analyzing business processes in a multi-faceted way.

Acknowledgment

We would like to thank the Flemish Research Council for financial support under Odysseus Grant B.0915.09. Also a special thanks to Karel Dejaeger for his computational and LATEX-support and to the Fluxicon-team, Anne Rozinat and Christian Günther, for their Nitro tool. Finally, we would like to express our gratitude to Stijn Goedertier for the fruitful discussions on process discovery and process mining in general.

Appendix A. Detailed accuracy results for the real-life event logs

Accuracy metrics for different data sets are shown in Tables A1–A8.

Table A1
Accuracy metrics for UFM data set.

Technique	Recall					Precision				Run time
	f	r_B^p	$PF_{Complete}$	PM	set	a'_B	s_B^n	p_B	etc_P	
AGNEsMiner	0.426	0.895	na	na	0.673	0.522	0.674	0.097	na	2 d:23 h:32 m:54 s
α^+	0.703	0.804	na	na	0.084	0.694	0.077	0.033	0.188	1 s
α^{++}	0.663	0.406	na	na	0.000	0.643	0.490	0.030	1.000	1 s
DT Genetic Miner	0.823	0.699	na	na	0.553	1.000	0.740	0.095	na	4 d:13 h:49 m:15 s
Flower	1.000	1.000	0.145	0.000	1.000	0.694	0.000	0.038	1.000	1 s
Genetic Miner	0.121	0.522	0.599	0.030	0.067	0.726	0.966	0.372	1.000	22 d:11 h:58 m:19 s
HeuristicsMiner	0.203	0.400	0.000	0.000	0.000	0.735	0.931	0.184	na	5 s
ILP Miner	1.000	1.000	na	na	1.000	0.564	0.407	0.062	0.541	5 m:36 s

Table A2
Accuracy metrics for P2P data set.

Technique	Recall					Precision				Run time
	f	r_B^p	$PF_{Complete}$	PM	set	a'_B	s_B^n	p_B	etc_P	
AGNEsMiner	0.893	0.980	na	na	0.000	1.000	0.965	0.624	0.706	23 m:32 s
α^+	0.832	0.729	na	na	0.000	0.326	0.801	0.181	0.158	1 s
α^{++}	na	na	na	na	na	na	na	na	na	1 s
DT Genetic Miner	0.982	0.982	0.000	0.000	0.970	0.594	0.948	0.533	na	1 d:6 h:2 m:26 s
Flower	1.000	1.000	0.571	0.000	1.000	0.420	0.000	0.057	0.125	1 s
Genetic Miner	0.753	0.879	0.652	0.000	0.110	0.742	0.980	0.724	na	1 d:8 h:4 m:49 s
HeuristicsMiner	0.996	0.875	0.662	0.000	0.973	0.856	0.987	0.803	0.550	4 s
ILP Miner	0.899	0.990	na	na	1.000	0.493	0.770	0.205	0.404	29 s

Table A3
Accuracy metrics for KHD data set.

Technique	Recall					Precision				Run time
	f	r_B^p	$PF_{Complete}$	PM	set	a'_B	s_B^n	p_B	etc_P	
AGNEsMiner	0.731	0.853	na	na	0.451	0.570	0.905	0.397	na	47 m:56 s
α^{+}	0.527	0.367	0.000	0.000	0.000	0.499	0.991	0.741	1.000	1 s
α^{++}	0.580	0.428	0.000	0.000	0.000	0.470	0.985	0.674	1.000	1 s
DT Genetic Miner	0.573	0.926	na	na	0.028	0.685	0.859	0.325	na	10 h:22 m:32 s
Flower	1.000	1.000	0.288	0.000	1.000	0.520	0.000	0.068	0.208	1 s
Genetic Miner	0.439	0.912	0.906	0.531	0.070	0.680	0.960	0.625	na	4 h:23 m:57 s
HeuristicsMiner	0.485	0.703	0.554	0.010	0.049	0.868	0.944	0.481	na	4 s
ILP Miner	0.933	1.000	na	na	1.000	0.749	0.513	0.131	0.371	26 s

Table A4
Accuracy metrics for SIM data set.

Technique	Recall					Precision				Run time
	f	r_B^p	$PF_{Complete}$	PM	set	a'_B	s_B^n	p_B	etc_P	
AGNEsMiner	0.758	0.847	na	na	0.065	0.675	0.891	0.285	1.000	50 m:40 s
α^{+}	0.586	0.336	na	na	0.000	0.438	0.819	0.088	1.000	1 s
α^{++}	0.694	0.309	na	na	0.000	0.500	0.881	0.118	1.000	1 s
DT Genetic Miner	0.465	0.830	na	na	0.022	0.693	0.898	0.297	na	16 h:50 m:18 s
Flower	1.000	1.000	0.659	0.000	1.000	0.540	0.000	0.049	0.110	1 s
Genetic Miner	0.288	0.873	0.935	0.540	0.001	0.731	0.882	0.276	na	6 h:15 m:14 s
HeuristicsMiner	0.794	0.703	0.226	0.000	0.000	1.000	0.898	0.262	na	1 s
ILP Miner	0.998	1.000	na	na	1.000	1.000	0.709	0.150	0.315	1 m:29 s

Table A5
Accuracy metrics for XBM data set.

Technique	Recall					Precision				Run time
	f	r_B^p	$PF_{Complete}$	PM	set	a'_B	s_B^n	p_B	etc_P	
AGNEsMiner	0.689	0.762	na	na	0.087	0.614	0.892	0.339	na	36 m:33 s
α^{+}	0.898	0.835	na	na	0.000	0.640	0.421	0.095	1	1 s
α^{++}	0.750	0.523	na	na	0.000	0.609	0.732	0.125	1	1 s
DT Genetic Miner	0.752	0.974	na	na	0.014	0.681	0.900	0.415	na	22 h:8 m:26 s
Flower	1.000	1.000	0.332	0.000	1.000	0.360	0.000	0.068	0.204	1 s
Genetic Miner	0.473	0.810	0.902	0.054	0.065	0.707	0.821	0.249	na	1 d:8 h:28 m:55 s
HeuristicsMiner	0.970	0.675	0.000	0.000	0.907	0.789	0.950	0.498	na	1 s
ILP Miner	0.980	0.792	na	na	1.000	0.834	0.856	0.286	0.605	53 s

Table A6
Accuracy metrics for XOA data set.

Technique	Recall					Precision				Run time
	f	r_B^p	$PF_{Complete}$	PM	set	a'_B	s_B^n	p_B	etc_P	
AGNEsMiner	0.765	0.862	na	na	0.666	0.363	0.964	0.344	na	3 h:24 m:17 s
α^{+}	0.927	0.702	na	na	0.065	0.598	0.971	0.346	0.424	2 s
α^{++}	0.750	0.614	0.526	0.034	0.000	0.598	0.992	0.630	1.000	1 s
DT Genetic Miner	0.795	0.894	0.846	0.000	0.506	0.431	0.950	0.280	na	5 h:33 m:44 s
Flower	1.000	1.000	0.355	0.000	1.000	0.100	0.000	0.022	0.060	1 s
Genetic Miner	0.684	0.940	0.829	0.026	0.467	0.707	0.970	0.407	na	5 h:9 m:45 s
HeuristicsMiner	0.985	0.840	0.784	0.137	0.836	0.856	0.995	0.776	na	1 s
ILP Miner	0.958	0.953	na	na	1.000	0.591	0.971	0.420	0.694	1.000

Table A7
Accuracy metrics for XNB data set.

Technique	Recall					Precision				Run time
	f	r_B^p	$PF_{Complete}$	PM	set	a'_B	s_B^n	p_B	etc_P	
AGNEsMiner	0.724	0.735	0.483	0.000	0.170	0.342	0.915	0.051	1.000	6 h:11 m:44 s
α^+	na	na	na	na	na	na	na	na	na	na
α^{++}	na	na	na	na	na	na	na	na	na	na
DT Genetic Miner	0.686	0.681	na	na	0.074	0.837	0.928	0.056	na	2 d:3 h:43 m:34 s
Flower	1.000	1.000	0.352	0.000	1.000	0.000	0.074	0.007	0.063	1 s
Genetic Miner	0.441	0.618	0.590	0.000	0.044	0.764	0.937	0.058	na	2 d:8 h:11 m:45 s
HeuristicsMiner	0.779	0.779	0.728	0.089	0.172	0.429	0.989	0.305	na	21 s
ILP Miner	0.977	0.940	na	na	1.000	0.115	0.978	0.215	0.751	6 m:7 s

Table A8
Accuracy metrics for XAO data set.

Technique	Recall					Precision				Run time
	f	r_B^p	$PF_{Complete}$	PM	set	a'_B	s_B^n	p_B	etc_P	
AGNEsMiner	0.967	0.953	0.953	0.785	0.785	0.635	0.974	0.763	0.734	4 m:21 s
α^+	0.710	0.638	na	na	0.117	0.737	0.920	0.412	0.706	1 s
α^{++}	0.745	0.708	0.576	0.020	0.117	0.645	0.913	0.417	1.000	1 s
DT Genetic Miner	0.936	0.983	0.971	0.085	0.899	0.654	0.949	0.628	na	1 h:11 m:58 s
Flower	1.000	1.000	0.324	0.000	1.000	0.180	0.000	0.081	0.162	1 s
Genetic Miner	0.995	0.999	0.960	0.741	0.995	0.648	0.966	0.721	na	1 h:2 m:22 s
HeuristicsMiner	0.986	0.853	0.834	0.114	0.811	0.694	0.983	0.817	na	1 s
ILP Miner	0.976	0.934	na	na	1.000	0.400	0.924	0.520	0.706	16 s

Appendix B. Detailed comprehensibility results for the real-life event logs

Comprehensibility metrics for different data sets are shown in Tables B1–B8.

Table B1
Comprehensibility metrics for UFM data set.

Technique	Transitions	Places	Arcs	AND-Joins	AND-Splits	OR-Joins	OR-Splits	ECaM	ECyM	SM
AGNEsMiner	79	131	2508	31	27	90	94	605	∞	16,590
α^+	42	4	10	0	0	1	1	3	153	87
α^{++}	42	20	180	28	28	17	17	58	62	528
DT Genetic Miner	212	120	476	25	16	18	19	298	∞	14,555
Flower	44	3	88	0	0	1	1	3	44	44
Genetic Miner	363	1633	5865	251	260	755	498	2701	∞	6,555,780
HeuristicsMiner	163	94	535	49	58	40	52	271	153,171	148,330
ILP Miner	42	329	6224	40	41	328	327	3111	∞	588

Table B2
Comprehensibility metrics for P2P data set.

Technique	Transitions	Places	Arcs	AND-Joins	AND-Splits	OR-Joins	OR-Splits	ECaM	ECyM	SM
AGNEsMiner	23	17	49	1	1	5	7	22	∞	464
α^+	23	19	74	11	7	9	9	31	∞	644
α^{++}	na	na	na	na	na	na	na	na	na	na
DT Genetic Miner	32	35	91	8	7	10	5	44	328	2688
Flower	25	3	50	0	0	1	1	3	25	25
Genetic Miner	47	35	110	11	5	10	11	56	165	6909
HeuristicsMiner	32	28	80	8	5	9	7	38	42	2605
ILP Miner	23	16	177	13	15	14	12	83	∞	966

Table B3
Comprehensibility metrics for KHD data set.

Technique	Transitions	Places	Arcs	AND-Joins	AND-Splits	OR-Joins	OR-Splits	ECaM	ECyM	SM
AGNEsMiner	36	27	123	23	13	9	12	63	7848	11,088
α^+	18	50	192	15	15	19	31	100	2	1638
α^{++}	18	66	318	15	15	37	57	176	3	2646
DT Genetic Miner	62	48	168	11	12	11	17	81	∞	6076
Flower	20	3	40	0	0	1	1	3	20	20
Genetic Miner	72	71	258	17	12	27	19	122	∞	26,462
HeuristicsMiner	55	33	153	26	17	16	15	81	∞	11,341
ILP Miner	18	26	406	16	12	25	25	109	∞	8705

Table B4
Comprehensibility metrics for SIM data set.

Technique	Transitions	Places	Arcs	AND-Joins	AND-Splits	OR-Joins	OR-Splits	ECaM	ECyM	SM
AGNEsMiner	60	42	158	6	0	8	5	26	2455	2646
α^+	22	55	187	15	17	30	17	76	∞	10,780
α^{++}	22	115	604	18	19	103	82	221	∞	13,552
DT Genetic Miner	131	227	605	76	68	52	44	293	184,713	74,277
Flower	24	3	48	0	0	1	1	3	24	24
Genetic Miner	81	57	211	8	12	23	15	106	0	22,401
HeuristicsMiner	42	29	102	6	8	11	11	46	∞	842
ILP Miner	22	24	341	14	20	22	22	122	∞	924

Table B5
Comprehensibility metrics for XBM data set.

Technique	Transitions	Places	Arcs	AND-Joins	AND-Splits	OR-Joins	OR-Splits	ECaM	ECyM	SM
AGNEsMiner	24	18	130	20	13	11	9	51	173	4032
α^+	18	11	29	4	3	3	4	16	∞	756
α^{++}	18	19	80	8	9	13	14	36	29	900
DT Genetic Miner	32	24	79	5	4	6	7	38	∞	896
Flower	20	3	40	0	0	1	1	3	0	20
Genetic Miner	54	59	196	24	17	16	22	99	∞	36,358
HeuristicsMiner	40	33	143	21	15	12	13	62	983	2583
ILP Miner	18	27	373	16	17	25	25	181	∞	126

Table B6
Comprehensibility metrics for XOA data set.

Technique	Transitions	Places	Arcs	AND-Joins	AND-Splits	OR-Joins	OR-Splits	ECaM	ECyM	SM
AGNEsMiner	64	41	166	18	14	12	20	71	12,004	25,375
α^+	49	96	513	25	38	57	26	137	82	1,088,476
α^{++}	49	152	996	32	41	115	91	453	40	147,001
DT Genetic Miner	51	40	128	5	10	10	5	54	9940	10,290
Flower	51	3	102	0	0	1	1	3	51	51
Genetic Miner	60	64	201	22	10	10	23	95	∞	67,432
HeuristicsMiner	75	60	236	24	36	21	24	95	508	48,777
ILP Miner	49	46	362	25	32	36	29	158	∞	18,817

Table B7
Comprehensibility metrics for XNB data set.

Technique	Transitions	Places	Arcs	AND-Joins	AND-Splits	OR-Joins	OR-Splits	ECaM	ECyM	SM
AGNEsMiner	178	115	392	4	32	18	22	na	na	na
α^+	na	na	na	na	na	na	na	na	na	na
α^{++}	na	na	na	na	na	na	na	na	na	na
DT Genetic Miner	224	149	529	33	27	16	25	na	na	na
Flower	165	3	336	0	0	1	1	3	165	165
Genetic Miner	269	347	1074	117	109	78	94	524	698	288,099
HeuristicsMiner	287	189	1999	79	87	72	72	na	na	na
ILP Miner	163	175	5096	114	123	139	137	3103	∞	30,996

Table B8
Comprehensibility metrics for XAO data set.

Technique	Transitions	Places	Arcs	AND-Joins	AND-Splits	OR-Joins	OR-Splits	ECaM	ECyM	SM
AGNEsMiner	16	13	48	5	6	8	4	21	41	1051
α^+	14	19	65	8	9	10	6	26	25	911
α^{++}	14	18	69	8	10	12	6	25	34	1912
DT Genetic Miner	30	19	63	1	2	6	6	31	166	120
Flower	16	3	32	0	0	1	1	3	16	16
Genetic Miner	30	21	73	6	6	7	9	36	∞	610
HeuristicsMiner	23	19	66	7	8	6	8	32	46	966
ILP Miner	14	11	84	5	10	7	8	28	∞	911

References

[1] W.M.P. van der Aalst, H.A. Reijers, A.J.M.M. Weijters, B.F. van Dongen, A.K. Alves de Medeiros, M. Song, H.M.W. Verbeek, Business process mining: an industrial application, Information Systems 32 (2007) 713–732.

[2] W.M.P. van der Aalst, A.J.M.M. Weijters, L. Maruster, Workflow mining: discovering process models from event logs, IEEE Transactions on Knowledge and Data Engineering 16 (2004) 1128–1142.

[3] P.-N. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, Addison-Wesley, 2005.

[4] M. Weske, Business Process Management: Concepts, Languages, Architectures, Springer, 2007.

[5] M. Song, W.M.P. van der Aalst, Towards comprehensive support for organizational mining, Decision Support Systems 46 (2008) 300–317.

[6] W.M.P. van der Aalst, M.H. Schonenberg, M. Song, Time prediction based on process mining, Information Systems 36 (2011) 450–475.

[7] W.M.P. van der Aalst, B.F. van Dongen, C.W. Günther, A. Rozinat, E. Verbeek, T. Weijters, ProM: the process mining toolkit, in: A.K.A. de Medeiros, B. Weber (Eds.), BPM (Demos), CEUR Workshop Proceedings, CEUR-WS.org, vol. 489, 2009.

[8] R. Agrawal, D. Gunopulos, F. Leymann, Mining process models from workflow logs, in: H. Schek, F. Saltor, I. Ramos, G. Alonso (Eds.), Proceedings of the 6th International Conference on Extending Database Technology (EDBT'98), Lecture Notes in Computer Science, vol. 1377, Springer, 1998, pp. 469–483.

[9] J. Cook, A. Wolf, Discovering models of software processes from event-based data, ACM Transactions on Software Engineering and Methodology 7 (1998) 215–249.

[10] A. Datta, Automating the discovery of AS-IS business process models: probabilistic and algorithmic approaches, Information Systems Research 9 (1998) 275–301.

[11] H. Mannila, C. Meek, Global partial orders from sequential data, in: Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '00), ACM, New York, NY, USA, 2000, pp. 161–168.

[12] G. Schimm, Process miner—a tool for mining process schemes from event-based data, in: S. Flesca, S. Greco, N. Leone, G. Ianni (Eds.), JELIA, Lecture Notes in Computer Science, vol. 2424, Springer, 2002, pp. 525–528.

[13] G. Schimm, Mining exact models of concurrent workflows, Computers in Industry 53 (2004) 265–281.

[14] A.K. Alves de Medeiros, B.F. van Dongen, W.M.P. van der Aalst, A.J.M.M. Weijters, Process Mining: Extending the Alpha-Algorithm to Mine Short Loops, BETA Working Paper Series 113, TU Eindhoven, 2004.

[15] L. Wen, W.M.P. van der Aalst, J. Wang, J. Sun, Mining process models with non-free-choice constructs, Data Mining and Knowledge Discovery 15 (2007) 145–180.

[16] L. Wen, J. Wang, W.M.P. van der Aalst, B. Huang, J. Sun, A novel approach for process mining based on event types, Journal of Intelligent Information Systems 32 (2009) 163–190.

[17] A.J.M.M. Weijters, W.M.P. van der Aalst, A.K. Alves de Medeiros, Process Mining with the HeuristicsMiner algorithm, BETA Working Paper Series 166, TU Eindhoven, 2006.

[18] C.W. Günther, W.M.P. van der Aalst, Fuzzy mining—adaptive process simplification based on multi-perspective metrics, in: G. Alonso, P. Dadam, M. Rosemann (Eds.), Proceedings of the 5th International Conference on Business Process Management, BPM 2007, Lecture Notes in Computer Science, vol. 4714, Brisbane, Australia, Springer, September 24–28, 2007, pp. 328–343.

[19] L. Maruster, A.J.M.M. Weijters, W.M.P. van der Aalst, A. van den Bosch, A rule-based approach for process discovery: dealing with noise and imbalance in process logs, Data Mining and Knowledge Discovery 13 (2006) 67–87.

[20] H. Ferreira, D. Ferreira, An integrated life cycle for workflow management based on learning and planning, International Journal of Cooperative Information Systems 15 (2006) 485–505.

[21] E. Lamma, P. Mello, M. Montali, F. Riguzzi, S. Storari, Inducing declarative logic-based models from labeled traces, in: G. Alonso, P. Dadam, M. Rosemann (Eds.), Proceedings of the 5th International Conference on Business Process Management, BPM 2007, Lecture Notes in Computer Science, vol. 4714, Brisbane, Australia, Springer, September 24–28, 2007, pp. 344–359.

[22] A.K. Alves de Medeiros, A.J.M.M. Weijters, W.M.P. van der Aalst, Genetic process mining: an experimental evaluation, Data Mining and Knowledge Discovery 14 (2007) 245–304.

[23] S. Goedertier, D. Martens, J. Vanthienen, B. Baesens, Robust process discovery with artificial negative events, Journal of Machine Learning Research 10 (2009) 1305–1340.

[24] H. Blockeel, L. De Raedt, Top-down induction of first-order logical decision trees, Artificial Intelligence 101 (1998) 285–297.

[25] G. Greco, A. Guzzo, L. Pontieri, D. Saccà, Discovering expressive process models by clustering log traces, IEEE Transactions on Knowledge and Data Engineering 18 (2006) 1010–1027.

[26] G. Greco, A. Guzzo, L. Pontieri, Mining taxonomies of process models, Data & Knowledge Engineering 67 (2008) 74–102.

[27] W.M.P. van der Aalst, V. Rubin, B.F. van Dongen, E. Kindler, C.W. Günther, Process Mining: A Two-Step Approach using Transition Systems and Regions, BPM-06-30, BPM Center Report, 2006.

[28] W.M.P. van der Aalst, V. Rubin, H.M.W. Verbeek, B.F. van Dongen, E. Kindler, C.W. Günther, Process mining: a two-step approach to balance between underfitting and overfitting, Software and System Modeling 9 (2010) 87–111.

[29] J. Carmona, J. Cortadella, M. Kishinevsky, New region-based algorithms for deriving bounded Petri nets, IEEE Transactions on Computers 59 (2010) 371–384.

[30] J. Herbst, D. Karagiannis, Workflow mining with InWoLvE, Computers in Industry 53 (2004) 245–264.

[31] B.F. van Dongen, W.M.P. van der Aalst, Multi-phase process mining: aggregating instance graphs into EPCs and Petri nets, in: Proceedings of the 2nd International Workshop on Applications of Petri Nets to Coordination, Workflow and Business Process Management (PNCWB).

[32] W. Gaaloul, K. Baïna, C. Godart, Towards mining structural workflow patterns, in: K.V. Andersen, J.K. Debenham, R. Wagner (Eds.), DEXA, Lecture Notes in Computer Science, vol. 3588, Springer, 2005, pp. 24–33.

[33] A.J.M.M. Weijters, W.M.P. van der Aalst, Rediscovering workflow models from event-based data using little thumb, Integrated Computer-Aided Engineering 10 (2003) 151–162.

[34] A.K. Alves de Medeiros, Genetic Process Mining, Ph.D. Thesis, TU Eindhoven, 2006.

[35] F. Folino, G. Greco, A. Guzzo, L. Pontieri, Discovering expressive process models from noised log data, in: B.C. Desai, D. Saccà, S. Greco (Eds.), IDEAS, ACM International Conference Proceeding Series, ACM, 2009, pp. 162–172.

[36] D.R. Ferreira, D. Gillblad, Discovering process models from unlabelled event logs, in: U. Dayal, J. Eder, J. Koehler, H.A. Reijers (Eds.), Proceedings of the 7th International Conference on Business Process Management, BPM 2009, Lecture Notes in Computer Science, vol. 5701, Ulm, Germany, Springer, September 8–10, 2009, pp. 143–158.

[37] J.M.E.M. van der Werf, B.F. van Dongen, C.A.J. Hurkens, A. Serebrenik, Process discovery using integer linear programming, Fundamenta Informaticae 94 (2009) 387–412.
[38] C.J. van Rijsbergen, Information Retrieval, Butterworth, 1979.
[39] A. Rozinat, M. Veloso, W.M.P. van der Aalst, Using Hidden Markov Models to Evaluate the Quality of Discovered Process Models, BPM Center Report BPM-08-10, 2008.
[40] A. Rozinat, W.M.P. van der Aalst, Conformance checking of processes based on monitoring real behavior, Information Systems 33 (2008) 64–95.
[41] A. Rozinat, Process Mining: Conformance and Extension, Ph.D. Thesis, TU Eindhoven, 2010.
[42] J. De Weerdt, M. De Backer, J. Vanthienen, B. Baesens, A robust F-measure for evaluating discovered process models, in: IEEE Symposium Series in Computational Intelligence (SSCI 2011), Paris, France, 2011.
[43] J. Munoz-Gama, J. Carmona, A fresh look at precision in process conformance, in: R. Hull, J. Mendling, S. Tai (Eds.), Business Process Management, Lecture Notes in Computer Science, vol. 6336, Springer, 2010, pp. 211–226.
[44] A.K. Alves de Medeiros, W.M.P. van der Aalst, A.J.M.M. Weijters, Quantifying process equivalence based on observed behavior, Data & Knowledge Engineering 64 (2008) 55–74.
[45] T. Calders, C.W. Günther, M. Pechenizkiy, A. Rozinat, Using minimum description length for process mining, in: S.Y. Shin, S. Ossowski (Eds.), SAC, ACM, 2009, pp. 1451–1455.
[46] J. De Weerdt, M. De Backer, J. Vanthienen, B. Baesens, A critical evaluation study of model-log metrics in process discovery, in: M. zur Muehlen, J. Su (Eds.), Business Process Management Workshops, Lecture Notes in Business Information Processing, vol. 66, Springer, 2010, pp. 158–169.
[47] S. Goedertier, J. De Weerdt, D. Martens, J. Vanthienen, B. Baesens, Process discovery in event logs: an application in the telecom industry, Applied Soft Computing 11 (2011) 1697–1710.
[48] C.W. Günther, Process Mining in Flexible Environments, Ph.D. Thesis, TU Eindhoven, 2009.
[49] J. Mendling, H.A. Reijers, J. Cardoso, What makes process models understandable? in: G. Alonso, P. Dadam, M. Rosemann (Eds.), Proceedings of the 5th International Conference on Business Process Management, BPM 2007, Lecture Notes in Computer Science, vol. 4714, Brisbane, Australia, Springer, September 24–28, 2007, pp. 48–63.
[50] K.B. Lassen, W.M.P. van der Aalst, Complexity metrics for workflow nets, Information & Software Technology 51 (2009) 610–626.
[51] J. Cardoso, Evaluating the process control-flow complexity measure, in: ICWS, IEEE Computer Society, 2005, pp. 803–804.
[52] T.J. McCabe, A complexity measure, IEEE Transactions on Software Engineering 2 (1976) 308–320.
[53] J. Dems˜ar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.
[54] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, The Annals of Mathematical Statistics 11 (1940) 86–92.
[55] O.J. Dunn, Multiple comparisons among means, Journal of the American Statistical Association 56 (1961) 52–64.
[56] I.T. Jolliffe, Principal Component Analysis, 2nd ed., Springer, 2002.
[57] H. Hotelling, Relations between two sets of variates, Biometrika 28 (1936) 321–377.
[58] C. Bratosin, N. Sidorova, W.M.P. van der Aalst, Discovering process models with genetic algorithms using sampling, in: R. Setchi, I. Jordanov, R.J. Howlett, L.C. Jain (Eds.), KES (1), Lecture Notes in Computer Science, vol. 6276, Springer, 2010, pp. 41–50.
[59] R. Cattell, Scree test for number of factors, Multivariate Behavioral Research 1 (1966) 245–276.
[60] D. Stewart, W. Love, A general canonical correlation index, Psychological Bulletin 70 (1968) 160–163.
[61] R.P.J.C. Bose, W.M.P. van der Aalst, Abstractions in process mining: a taxonomy of patterns, in: U. Dayal, J. Eder, J. Koehler, H.A. Reijers (Eds.), Proceedings of the 7th International Conference on Business Process Management, BPM 2009, Lecture Notes in Computer Science, vol. 5701, Ulm, Germany, Springer, 2009, September 8–10, pp. 159–175.
[62] M. Song, C.W. Günther, W.M.P. van der Aalst, Trace clustering in process mining, in: D. Ardagna, M. Mecella, J. Yang (Eds.), Business Process Management Workshops, Lecture Notes in Business Information Processing, vol. 17, Springer, 2008, pp. 109–120.
[63] D.R. Ferreira, M. Zacarias, M. Malheiros, P. Ferreira, Approaching process mining with sequence clustering: experiments and findings, in: G. Alonso, P. Dadam, M. Rosemann (Eds.), Proceedings of the 5th International Conference on Business Process Management, BPM 2007, Lecture Notes in Computer Science, vol. 4714, Brisbane, Australia, Springer, September 24–28, 2007, pp. 360–374.
[64] W.M.P. van der Aalst, M. Pesic, H. Schonenberg, Declarative workflows: balancing between flexibility and support, Computer Science—R&D 23 (2009) 99–113.
[65] W.M.P. van der Aalst, Process Mining—Discovery, Conformance and Enhancement of Business Processes, Springer, 2011.
[66] W. van der Aalst, The application of petri nets to workflow management, Journal of Circuits, Systems, and Computers 8 (1998) 21–66.
[67] R.P.J.C. Bose, W.M.P. van der Aalst, Process diagnostics using trace alignment: opportunities, issues, and challenges, Information Systems 37 (2012) 117–141.