Automated Discovery of Process Models from Event Logs: Review and Benchmark

Adriano Augusto, Raffaele Conforti, Marlon Dumas, Marcello La Rosa, Fabrizio Maria Maggi, Andrea Marrella, Massimo Mecella, and Allar Soo

Abstract—Process mining allows analysts to exploit logs of historical executions of business processes to extract insights regarding the actual performance of these processes. One of the most widely studied process mining operations is automated process discovery. An automated process discovery method takes as input an event log, and produces as output a business process model that captures the control-flow relations between tasks that are observed in or implied by the event log. Various automated process discovery methods have been proposed in the past two decades, striking different tradeoffs between scalability, accuracy, and complexity of the resulting models. However, these methods have been evaluated in an ad-hoc manner, employing different datasets, experimental setups, evaluation measures, and baselines, often leading to incomparable conclusions and sometimes unreproducible results due to the use of closed datasets. This article provides a systematic review and comparative evaluation of automated process discovery methods, using an open-source benchmark and covering 12 publicly-available real-life event logs, 12 proprietary real-life event logs, and nine quality metrics. The results highlight gaps and unexplored tradeoffs in the field, including the lack of scalability of some methods and a strong divergence in their performance with respect to the different quality metrics used.

Index Terms—Process mining, automated process discovery, survey, benchmark

1 Introduction

Modern information systems maintain detailed trails of the business processes they support, including records of key process execution events, such as the creation of a case or the execution of a task within an ongoing case. Process mining techniques allow analysts to extract insights about the actual performance of a process from collections of such event records, also known as *event logs* [1]. In this context, an event log consists of a set of traces, each trace itself consisting of the sequence of events related to a given case.

One of the most widely investigated process mining operations is *automated process discovery*. An automated process discovery method takes as input an event log, and produces as output a *business process model* that captures the control-flow relations between tasks that are observed in or implied by the event log.

In order to be useful, such automatically discovered process models must accurately reflect the behavior recorded in the log. Specifically, the process model discovered from an event log should be able to: (i) generate each trace in the log, or for each trace in the log, generate a trace that is similar to it; (ii) generate traces that are not in the log but that are identical or similar to traces of the process that produced the log; and (iii) not generate other traces [2]. The first property is called *fitness*, the second *generalization* and the third *precision*. In addition, the discovered process model should be as simple as possible, a property that is usually quantified via *complexity* measures.

The problem of automated discovery of process models from event logs has been intensively researched in the past two decades. Despite a rich set of proposals, state-of-the-art automated process discovery methods suffer from two recurrent deficiencies when applied to real-life logs [3]: (i) they produce large and spaghetti-like models; and (ii) they produce models that either poorly fit the event log (low fitness) or over-generalize it (low precision or low generalization). Striking a tradeoff between these quality dimensions in a robust manner has proved to be a difficult problem.

So far, automated process discovery methods have been evaluated in an ad hoc manner, with different authors employing different datasets, experimental setups, evaluation measures and baselines, often leading to incomparable conclusions and sometimes unreproducible results due to the use of non-publicly available datasets. This work aims at filling this gap by providing: (i) a systematic review of automated process discovery methods; and (ii) a comparative evaluation of seven implementations of representative methods, using an open-source benchmark framework and covering twelve publicly-available real-life event logs, twelve proprietary real-life event logs, and nine quality metrics covering all four dimensions mentioned above (fitness, precision, generalization and complexity), as well as execution time.

The outcomes of this research are a classified inventory of automated process discovery methods and a benchmark

- *A. Augusto is with the University of Tartu, Tartu 50090, Estonia, and the University of Melbourne, Parkville, VIC 3010, Australia. E-mail: adriano.augusto@ut.ee.*
- *R. Conforti and M. La Rosa are with the University of Melbourne, Parkville, VIC 3010, Australia. E-mail: {raffaele.conforti, marcello.larosa}@unimelb.edu.au.*
- *M. Dumas, F.M. Maggi, and A. Soo are with the University of Tartu, Tartu 50090, Estonia. E-mail: {marlon.dumas, f.m.maggi}@ut.ee, allar.soo@gmail.com.*
- *A. Marrella and M. Mecella are with the Sapienza Universitá di Roma, Roma 00185, Italy. E-mail: {marrella, mecella}@diag.uniroma1.it.*

Manuscript received 12 May 2017; revised 22 Apr. 2018; accepted 20 May 2018. Date of publication 29 May 2018; date of current version 5 Mar. 2019. (Corresponding author: Marcello La Rosa.)
Recommended for acceptance by R. Gemulla.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2018.2841877

designed to enable researchers to empirically compare new automated process discovery methods against existing ones in a unified setting. The benchmark is provided as an open-source command-line Java application to enable researchers to replicate the reported experiments with minimal configuration effort.

The rest of the article is structured as follows. Section 2 describes the search protocol used for the systematic literature review, whereas Section 3 classifies the methods identified in the review. Then, Section 4 introduces the experimental benchmark and results, whereas Section 5 discusses the overall findings and Section 6 acknowledges the threats to the validity of the study. Finally, Section 7 relates this work to previous reviews and comparative studies in the field and Section 8 concludes the paper and outlines future work directions.

2 Search Protocol

In order to identify and classify research in the area of automated process discovery, we conducted a *Systematic Literature Review* (SLR) through a scientific, rigorous and replicable approach as specified by Kitchenham [4].

First, we formulated a set of research questions to scope the search, and developed a list of search strings. Next, we ran the search strings on different data sources. Finally, we applied inclusion criteria to select the studies retrieved through the search.

2.1 Research Questions

The objective of our SLR is to analyze research studies addressing the problem of automated (business) process discovery from event logs. Studies related to event log filtering, enhancement, and decomposition are orthogonal to automated process discovery. In this study, we focus only on the automated discovery problem. We consider that reviewing and comparing existing orthogonal studies (e.g., event log pre-processing [5], [6]) deserves a separate study. With this aim, we formulated the following research questions:

- RQ1 What methods exist for *automated* (business) process discovery from *event logs*?
- RQ2 What *type of process models* can be discovered by these methods, and in which *modeling language*?
- RQ3 Which *semantics* can be captured by a model discovered by these methods?
- RQ4 What *tools* are available to support these methods?
- RQ5 What *type of data* has been used to evaluate these methods, and from which application domains?

RQ1 is the core research question, which aims at identifying existing methods to perform (business) process discovery from event logs. The other questions allow us to identify a set of classification criteria. Specifically, RQ2 categorizes the output of a method on the basis of the type of process model discovered (i.e., procedural, declarative or hybrid), and the specific modeling language employed (e.g., Petri nets, BPMN, Declare). RQ3 delves into the specific semantic constructs supported by a method (e.g., exclusive choice, parallelism, loops). RQ4 explores what tool support the different methods have, while RQ5 investigates how the methods have been evaluated and in which application domains.

2.2 Search String Development and Validation

Next, we developed four search strings by deriving keywords from our knowledge of the subject matter. We first determined that the term "process discovery" is a very generic term that would allow us to retrieve the majority of methods in this area. Furthermore, we used "learning" and "workflow" as synonyms of "discovery" and "process" (respectively). This led to the following four search strings: (i) "process discovery"; (ii) "workflow discovery"; (iii) "process learning"; and (iv) "workflow learning". We intentionally excluded the terms "automated" and "automating" in the search strings, because these terms are often not explicitly used.

However, this led to retrieving many more studies than those that actually focus on automated process discovery, e.g., studies on process discovery via workshops or interviews. Thus, if a query on a specific data source returned more than one thousand results, we refined it by combining the selected search string with the term "business" or "process mining" to obtain more focused results, e.g., "process discovery AND process mining" or "process discovery AND business". According to this criterion, the final search strings used for our search were the following:

i. "process discovery AND process mining"
ii. "process learning AND process mining"
iii. "workflow discovery"
iv. "workflow learning"

First, we applied each of the four search strings to Google Scholar, retrieving studies based on the occurrence of one of the search strings in the title, the keywords or the abstract of a paper. Then, we used the following six popular academic databases: Scopus, Web of Science, IEEE Xplore, ACM Digital Library, SpringerLink, ScienceDirect, to double check the studies retrieved from Google Scholar. We noted that this additional search did not return any relevant study that was not already discovered in our primary search. The search was completed in December 2017.

2.3 Study Selection

As a last step, as suggested by [7], [8], [9], [10], we defined inclusion criteria to ensure an unbiased selection of relevant studies. To be retained, a study must satisfy all the following inclusion criteria.

- IN1 *The study proposes a method for automated (business) process discovery from event logs*. This criterion draws the borders of our search scope and is direct consequence of RQ1.
- IN2 *The study proposes a method that has been implemented and evaluated*. This criterion let us exclude methods whose properties have not been evaluated nor analyzed.
- IN3 *The study is published in 2011 or later*. Earlier studies have been reviewed and evaluated by De Weerdt et al. [3], therefore, we decided to focus only on the successive studies. Nevertheless, we performed a mapping of the studies assessed in 2011 [3] and their successors (when applicable), cf. Table 1.
- IN4 *The study is peer-reviewed*. This criterion guarantees a minimum reasonable quality of the studies included in this SLR.
- IN5 *The study is written in English*.

Inclusion criteria IN3, IN4 and IN5 were automatically applied through the configuration of the search engines. After the application of the latter three inclusion criteria, we obtained a total of 2,820 studies. Then, we skimmed title and abstract of these studies to exclude those studies that were clearly not compliant with IN1. As a result of this first iteration, we obtained 344 studies.

TABLE 1
Methods Assessed by De Weerdt et al. [3] (Left) and the Respective Successors (Right)

α, α^+, α^{++} [11], [12], [13]	α\$ [14]
AGNEs Miner [15]	—
(DT) Genetic Miner [16], [17]	Evolutionary Tree Miner [18]
Heuristics Miner [19], [20]	Heuristics Miner [21], [22], [23], [24]
ILP Miner [25]	Hybrid ILP Miner [26]

Then, we assessed each of the 344 studies against the inclusion criteria IN1 and IN2. The (combined) assessment of IN1 and IN2 on the 344 selected studies was performed independently by two authors of this paper, whose decisions were compared in order to resolve inconsistencies with the mediation of a third author, when needed. The assessment of IN1 was based on the accurate reading of the abstract, introduction and conclusion. On the other hand, to determine whether a study fulfilled IN2, we relied on the findings reported in the evaluation of the studies. As a result of these iterations, we found 86 studies matching the five inclusion criteria.

However, many of these studies refer to the same automated process discovery method, i.e., some studies are either extensions, optimization, preliminaries or generalization of another study. For such reason, we decided to group the studies by either the last version or the most general one. When in doubt, the grouping decision was taken after a consultation with the main author. At the end of this process, as shown in Table 2, 35 main groups of discovery algorithms were identified.

The excel sheet available at https://drive.google.com/open?id=1fW8WLXSwI2ntiPu3XVgsDI1cJUJr762G reports the 344 studies found after the first iteration. For each of these studies, we explicitly refer to the inclusion criterion violated for the study to be discarded. Furthermore, each selected study has a reference to the group it belongs to (unless it is the main study).

Fig. 1 shows how the studies are distributed over time. We can see that the interest in the topic of automated process discovery grew over time with a sharp increase between 2013 and 2014, and lately declining close to the average number of studies per year.

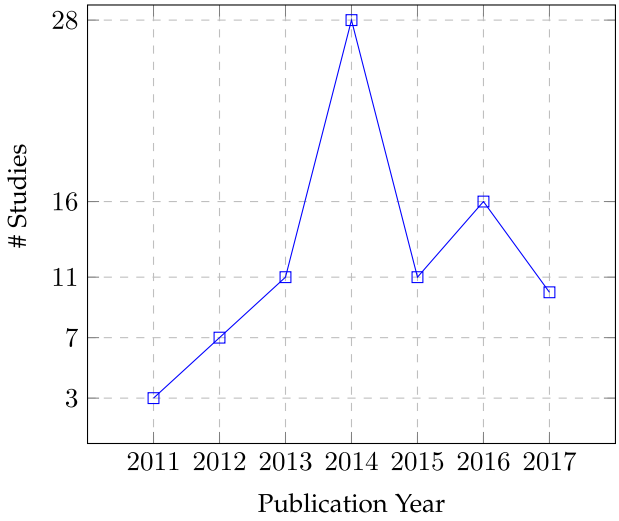

Fig. 1. Number of studies over time.

3 Classification of Methods

Driven by the research questions defined in Section 2.1, we identified the following classification dimensions to survey the methods described in the primary studies:

(1) Model type (procedural, declarative, hybrid) and model language (e.g., Petri nets, BPMN, Declare)—RQ2
(2) Semantic captured in procedural models: parallelism (AND), exclusive choice (XOR), inclusive choice (OR), loop—RQ3
(3) Type of implementation (standalone or plug-in, and tool accessibility)—RQ4
(4) Type of evaluation data (real-life, synthetic or artificial log, where a synthetic log is one generated from a real-life model while an artificial log is one generated from an artificial model) and application domain (e.g., insurance, banking, healthcare)—RQ5.

This information is summarized in Table 2. Each entry in this table refers to the main study of the 35 groups found. Also, we cited all the studies that relate to the main one. Collectively, the information reported by Table 2 allows us

TABLE 2
Overview of the 35 Primary Studies Resulting from the Search (Ordered by Year and Author)

Method	Main study	Year	Related studies	Model type	Model language	Semantic Constructs				Implementation		Evaluation		
						AND	XOR	OR	Loop	Framework	Accessible	Real-life	Synth.	Art.
HK	Huang and Kumar [27]	2012		Procedural	Petri nets	✓	✓		✓	Standalone		✓	✓	
Declare Miner	Maggi et al. [28]	2012	[29], [30], [31], [32], [33], [34]	Declarative	Declare					ProM	✓	✓	✓	
MINERful	Di Ciccio, Mecella [35]	2013	[36], [37], [38], [39]	Declarative	Declare					ProM, Standalone	✓	✓	✓	
Inductive Miner - Infrequent	Leemans et al. [40]	2013	[41], [42], [43], [44], [45], [46], [47]	Procedural	Process trees	✓	✓	✓	✓	ProM	✓	✓		
Data-aware Declare Miner	Maggi et al. [48]	2013		Declarative	Declare					ProM	✓	✓		
Process Skeletonization	Abe, Kudo [49]	2014	[50]	Procedural	Directly-follows graphs		✓		✓	Standalone		✓		
Evolutionary Declare Miner	vanden Broucke et al. [51]	2014		Declarative	Declare					Standalone			✓	
Evolutionary Tree Miner	Buijs e al. [18]	2014	[52], [53], [54], [55], [56]	Procedural	Process trees	✓	✓	✓	✓	ProM	✓	✓		✓
Aim	Carmona, Cortadella [57]	2014		Procedural	Petri nets	✓	✓		✓	Standalone		✓		✓
WoMan	Ferilli [58]	2014	[59], [60], [61], [62], [63]	Declarative	WoMan					Standalone		✓		✓
Hybrid Miner	Maggi et al. [64]	2014		Hybrid	Declare + Petri nets					ProM	✓	✓		
Competition Miner	Redlich et al. [65]	2014	[66], [67], [68]	Procedural	BPMN	✓	✓		✓	Standalone		✓		✓
Direted Acyclic Graphs	Vasilecas et al. [69]	2014		Procedural	Directed acyclic graphs		✓			Standalone		✓	✓	
Fusion Miner	De Smedt et al. [70]	2015		Hybrid	Declare + Petri nets					ProM	✓	✓	✓	✓
CNMining	Greco et al. [71]	2015	[72]	Procedural	Causal nets	✓	✓		✓	ProM	✓	✓	✓	
alpha\$	Guo et al. [14]	2015		Procedural	Petri nets	✓	✓		✓	ProM	✓	✓		✓
Maximal Pattern Mining	Liesaputra et al. [73]	2015		Procedural	Causal nets	✓	✓		✓	ProM		✓		✓
DGEM	Molka et al. [74]	2015		Procedural	BPMN	✓	✓		✓	Standalone		✓		✓
ProDiGen	Vazquez et al. [75]	2015	[76]	Procedural	Causal nets	✓	✓		✓	ProM			✓	✓
Non-Atomic Declare Miner	Bernardi et al. [77]	2016	[78]	Declarative	Declare					ProM	✓	✓	✓	
RegPFA	Breuker et al. [79]	2016	[80]	Procedural	Petri nets	✓	✓		✓	Standalone	✓	✓	✓	
BPMN Miner	Conforti et al. [81]	2016	[82]	Procedural	BPMN	✓	✓	✓	✓	Apromore, Standalone	✓	✓		✓
CSMMiner	van Eck et al. [83]	2016	[84]	Procedural	State machines	✓	✓		✓	ProM	✓	✓		
TAU miner	Li et al. [85]	2016		Procedural	Petri nets	✓	✓		✓	ProM		✓		✓
PGminer	Mokhov et al. [86]	2016		Procedural	Partial order graphs	✓	✓			Standalone, Workcraft	✓	✓		✓
SQLMiner	Schönig et al. [87]	2016	[88]	Declarative	Declare					Standalone	✓	✓		
ProM-D	Song et al. [89]	2016		Procedural	Petri nets	✓	✓		✓	Standalone		✓	✓	
CoMiner	Tapia-Flores et al. [90]	2016		Procedural	Petri nets	✓	✓		✓	ProM				✓
Proximity Miner	Yahya et al. [91]	2016	[92]	Procedural	Causal nets	✓	✓		✓	ProM	✓	✓		
Heuristics Miner	Augusto et al. [21]	2017	[22], [23], [24], [93]	Procedural	BPMN	✓	✓		✓	Apromore, Standalone	✓	✓	✓	✓
Split miner	Augusto et al. [94]	2017		Procedural	BPMN	✓	✓		✓	Apromore, Standalone	✓	✓		
Fodina	vanden Broucke et al. [95]	2017	[96]	Procedural	BPMN	✓	✓		✓	ProM	✓	✓	✓	
Stage miner	Nguyen et al. [97]	2017		Procedural	Causal nets	✓	✓		✓	Apromore, Standalone	✓	✓		
Decomposed Process Miner	Verbeek, van der Aalst [98]	2017	[99], [100], [101], [102], [103]	Procedural	Petri nets	✓	✓	✓	✓	ProM	✓	✓		✓
HybridILPMiner	van Zelst et al. [26]	2017	[104], [105]	Procedural	Petri nets	✓	✓		✓	ProM	✓			✓

to answer the first research question: "what methods exist for automated process discovery?".

In the remainder of this section, we proceed with surveying each main study method along the above classification dimensions, to answer the other research questions.

3.1 Model Type and Language (RQ2)

The majority of methods (26 out of 35) produce procedural models. Six approaches [28], [35], [48], [51], [77], [87] discover declarative models in the form of Declare constraints, whereas [58] produces declarative models using the WoMan formalism. The methods in [64], [70] are able to discover hybrid models as a combination of Petri nets and Declare constraints.

Regarding the modeling languages of the discovered process model, we notice that Petri nets is the predominant one. However, more recently, we have seen the appearance of methods that produce models in BPMN, a language that is more practically-oriented and less technical than Petri nets. This denotes a shift in the target audience of these methods, from data scientists to practitioners, such as business analysts and decision managers. Other technical languages employed, besides Petri nets, include Causal nets, State machines and simple Directed Acyclic Graphs, while Declare is the most commonly-used language when producing declarative models.

Petri Nets. In [27], the authors describe an algorithm to extract block-structured Petri nets from event logs. The algorithm works by first building an adjacency matrix between all pairs of tasks and then analyzing the information in it to extract block-structured models consisting of basic sequence, choice, parallel, loop, optional and self-loop structures as building blocks. The method has been implemented in a standalone tool called HK.

The method presented in [14] is based on the $\alpha\$$ algorithm, which can discover invisible tasks involved in non-free-choice constructs. The algorithm is an extension of the well-known α algorithm, one of the very first algorithms for automated process discovery, originally presented in [1].

In [98], the authors propose a generic divide-and-conquer framework for the discovery of process models from large event logs. The method partitions the event log into smaller logs and discovers a model from each of them. The output is then assembled from all the models discovered from the sub-logs. This method aims at producing high quality models by reducing the overall complexity. A range of preliminary studies [99], [100], [101], [102], [103] widely illustrate the idea of splitting a large event log into a collection of smaller logs to improve the performance of a discovery algorithm.

van Zelst et al. [26], [104], [105] propose an improvement of the ILP miner implemented in [25]. Their method is based on hybrid variable-based regions. Through hybrid variable-based regions, it is possible to vary the number of variables used within the ILP problems being solved. Using a different number of variables has an impact on the average computation time for solving the ILP problem.

In [79], [80], the authors propose an approach to discover Petri nets using the theory of grammatical inference. The method has been implemented as a standalone application called RegPFA.

The approach proposed in [89] is based on the observation that activities with no dependencies in an event log can be executed in parallel. In this way, this method can discover process models with concurrency even if the logs fail to meet the completeness criteria. The method has been implemented in a tool called ProM-D.

In [57], the authors propose the use of numerical abstract domains for discovering Petri nets from large logs while guaranteeing formal properties of the discovered models. The technique guarantees the discovery of Petri nets that can reproduce every trace in the log and that are minimal in describing the log behavior.

The approach introduced in [90] addresses the problem of discovering sound Workflow nets from incomplete logs. The method is based on the concept of invariant occurrence between activities, which is used to identify sets of activities (named conjoint occurrence classes) that can be used to infer the behaviors not exhibited in the log.

In [85], the authors leverage data carried by tokens in the execution of a business process to track the state changes in the so-called token logs. This information is used to improve the performance of standard discovery algorithms.

Process Trees. The Inductive Miner [40] and the Evolutionary Tree Miner [18] are both based on the extraction of process trees from an event log. Concerning the former, many different variants have been proposed during the last years, but its first appearance is in [41]. Successively, since that method was unable to deal with infrequent behavior, an improvement was proposed in [40], which efficiently drops infrequent behavior from logs, still ensuring that the discovered model is behaviorally correct (sound) and highly fitting. Another variant of the Inductive Miner is presented in [42]. This variant can minimize the impact of incompleteness of the input logs. In [44], the authors discuss ways of systematically treating lifecycle information in the discovery task. They introduce a process discovery technique that is able to handle lifecycle data to distinguish between concurrency and interleaving. The method proposed in [43] provides a graphical support for navigating the discovered model and the one described in [47] can deal with cancelation or error-handling behaviors (i.e., with logs containing traces that do not complete normally). Finally, the variant presented in [45] and [46] combines scalability with quality guarantees. It can be used to mine large event logs and produces sound models.

In [18], Buijs et al. introduce the Evolutionary Tree Miner. This method is based on a genetic algorithm that allows the user to drive the discovery process based on preferences with respect to the four quality dimensions of the discovered model: fitness, precision, generalization and complexity. The importance of these four dimensions and how to address their balance in process discovery is widely discussed in the related studies[52], [53], [54], [55], [56].

Causal Nets. Greco et al. propose a discovery method that returns causal nets [71], [72]. A causal net is a net where only the causal relation between activities in a log is captured. This method encodes causal relations gathered from an event log and, if available, background knowledge in terms of precedence constraints over the topology of the resulting model. A discovery algorithm is formulated in terms of reasoning problems over precedence constraints.

In [73], the authors propose a method for automated process discovery using Maximal Pattern Mining where they discover recurrent sequences of events in the traces of the log. Starting from these patterns they build process models in the form of causal nets.

ProDiGen, a standalone miner by Vazquez et al. [75], [76], allows users to discover causal nets from event logs using

a genetic algorithm. The algorithm is based on a fitness function that takes into account completeness, precision and complexity and specific crossover and mutation operators.

Another method that produces causal nets is the Proximity Miner, presented in [91], [92]. This method extracts behavioral relations between the events of the log, which are then enhanced using inputs from domain experts.

Finally, in [97], the authors propose a method to discover causal nets that optimizes the scalability and interpretability of the outputs. The process under analysis is decomposed into a set of stages, such that each stage can be mined separately. The proposed technique discovers a stage decomposition that maximizes modularity.

State Machines. The CSM Miner, discussed in [83], [84], discovers state machines from event logs. Instead of focusing on the events or activities that are executed in the context of a particular process, this method concentrates on the states of the different process perspectives and discover how they are related with each other. These relations are expressed in terms of Composite State Machines. The CSM Miner provides an interactive visualization of these multi-perspective state-based models.

BPMN Models. In [82], Conforti et al. present the BPMN Miner, a method for the automated discovery of BPMN models containing sub-processes, activity markers such as multi-instance and loops, and interrupting and non-interrupting boundary events (to model exception handling). The method has been subsequently improved in [81] to make it robust to noise in event logs.

Another method to discover BPMN models has been presented in [74]. In this approach, a hierarchical view on process models is formally specified and an evolution strategy is applied on it. The evolution strategy, which is guided by the diversity of the process model population, efficiently finds the process models that best represent a given event log.

A further method to discover BPMN models is the Dynamic Constructs Competition Miner [65], [67], [68]. This method extends the Constructs Competition Miner presented in [66]. The method is based on a divide-and-conquer algorithm that discovers block-structured process models from logs.

In [94], the authors propose a discovery method that produces simple process models with low branching complexity and consistently high and balanced fitness, precision and generalization. The approach combines a technique to filter the directly-follows graph induced by an event log, with an approach to identify combinations of split gateways that accurately capture the concurrency, conflict and causal relations between neighbors in the directly-follows graph.

Fodina [95], [96] is a discovery method based on the Heuristics Miner [20]. However, differently from the latter, Fodina is more robust to noisy data, is able to discover duplicate activities, and allows for flexible configuration options to drive the discovery according to end user inputs.

In [23], the authors present the Flexible Heuristics Miner. This method can discover process models containing non-trivial constructs but with a low degree of block structuredness. At the same time, the method can cope well with noise in event logs. The discovered models are a specific type of Heuristics nets where the semantics of splits and joins is represented using split/join frequency tables. This results in easy to understand process models even in the presence of non-trivial constructs and log noise. The discovery algorithm is based on that of the original Heuristics Miner method [20]. In [24], the method presented in [23] has been improved as anomalies were found concerning the validity and completeness of the resulting process model. The improvements have been implemented in the Updated Heuristics Miner. A data-aware version of the Heuristics Miner that takes into consideration data attached to events in a log has been presented in [22]. Finally, in [21], [93], the authors propose an improvement of the Heuristics Miner algorithm to separate the objective of producing accurate models and that of ensuring their structuredness and soundness. Instead of directly discovering a structured process model, the approach first discovers accurate, possibly unstructured (and unsound) process models, and then transforms the resulting process model into a structured (and sound) one.

Declarative Models. In [29], the authors present the first basic approach for mining declarative process models expressed using Declare constraints [106], [107]. This approach was improved in [28] using a two-phase approach. The first phase is based on an apriori algorithm used to identify frequent sets of correlated activities. A list of candidate constraints is built on the basis of the correlated activity sets. In the second phase, the constraints are checked by replaying the log on specific automata, each accepting only those traces that are compliant to one constraint. Those constraints satisfied by a percentage of traces higher than a user-defined threshold, are discovered. Other variants of the same approach are presented in [30], [31], [32], [33], [34]. The technique presented in [30] leverages apriori knowledge to guide the discovery task. In [31], the approach is adapted to be used in cross-organizational environments in which different organizations execute the same process in different variants. In [32], the author extends the approach to discover metric temporal constraints, i.e., constraints taking into account the time distance between events. Finally, in [33], [34], the authors propose mechanisms to reduce the execution times of the original approach presented in [28].

MINERful [35], [36], [37] discovers Declare constraints using a two-phase approach. The first phase computes statistical data describing the occurrences of activities and their interplay in the log. The second one checks the validity of Declare constraints by querying such a statistic data structure (knowledge base). In [38], [39], the approach is extended to discover target-branched Declare constraints, i.e., constraints in which the target parameter is the disjunction of two or more activities.

The approach presented in [48] is the first approach for the discovery of Declare constraints with an extended semantics that take into consideration data conditions. The data-aware semantics of Declare presented in this paper is based on first-order temporal logic. The method presented in [77], [78] is based on the use of discriminative rule mining to determine how the characteristics of the activity lifecycles in a business process influence the validity of a Declare constraint in that process.

Other approaches for the discovery of Declare constraints have been presented in [51], [87]. In [51], the authors present the Evolutionary Declare Miner that implements the discovery task using a genetic algorithm. The SQLMiner, presented in [87], is based on a mining approach that directly works on relational event data by querying a log with standard SQL. By leveraging database performance technology, the mining procedure is extremely fast. Queries can be customized and cover process perspectives beyond control flow [88].

The WoMan framework, proposed by Ferilli in [58] and further extended in the related studies [59], [60], [61], [62],

[63], includes a method to learn and refine process models from event logs, by discovering first-order logic constraints. It guarantees incrementality in learning and adapting the models, the ability to express triggers and conditions on the process tasks and efficiency.

Further Approaches. In [49], [50], the authors introduce a monitoring framework for automated process discovery. A monitoring context is used to extract traces from relational event data and attach different types of metrics to them. Based on these metrics, traces with certain characteristics can be selected and used for the discovery of process models expressed as directly-follows graphs.

Vasilecas et al. [69] present a method for the extraction of directed acyclic graphs from event logs. Starting from these graphs, they generate Bayesian belief networks, one of the most common probabilistic models, and use these networks to efficiently analyze business processes.

In [86], the authors show how conditional partial order graphs, a compact representation of families of partial orders, can be used for addressing the problem of compact and easy-to-comprehend representation of event logs with data. They present algorithms for extracting both the control flow as well as relevant data parameters from a given event log and show how conditional partial order graphs can be used to visualize the obtained results. The method has been implemented as a Workcraft plug-in and as a standalone application called PGminer.

The Hybrid Miner, presented in [64], puts forward the idea of discovering a hybrid model from an event log based on the semantics defined in [108]. According to such semantics, a hybrid process model is a hierarchical model, where each node represents a sub-process, which may be specified in a declarative or procedural way. Petri nets are used for representing procedural sub-processes and Declare for representing declarative sub-processes.

De Smedt et al., [70] proposes an approach for the discovery of hybrid models based on the semantics proposed in [109]. Differently from the semantics introduced in [108], where a hybrid process model is hierarchical, the semantics defined in [109] is devoted to obtain a fully mixed language, where procedural and declarative constructs can be connected with each other.

3.2 Procedural Language Constructs (RQ3)

All the 26 methods that discover a procedural model can detect the basic control-flow structure of sequence. Out of these methods, only four can also discover inclusive choices, but none in the context of non-block-structured models. In fact, [18], [40] are able to directly identify block-structured inclusive choices (using process trees), while [81], [98] can detect this construct only when used on top of the methods in [18] or [40] (i.e., indirectly).

The remaining 22 methods can discover constructs for parallelism, exclusive choice and loops, with the exception of [49], which can detect exclusive choice and loops but not parallelism, [86], which can detect parallelism and exclusive choice but not loops, and [69], which can discover exclusive choices only.

3.3 Implementation (RQ4)

Over 50 percent of the methods (19 out of 35) provide an implementation as a plug-in for the ProM platform.[1] The reason behind the popularity of ProM can be explained by its open-source and portable framework, which allows researchers to easily develop and test new discovery algorithms. Also, ProM is the first software tool for process mining. One of the methods that has a ProM implementation [35] is also available as standalone tool. The works [21], [81], [94], [97] provide both a standalone implementation and a further implementation as a plug-in for Apromore.[2] Apromore is an online process analytics platform also available under an open-source license. Finally, one method [86] has been implemented as a plug-in for Workcraft,[3] a platform for designing concurrent systems.

Notice that 22 tools out of 35 are made publicly available to the community. These exclude 4 ProM plug-ins and 9 standalone tools.

3.4 Evaluation Data and Domains (RQ5)

The surveyed methods have been evaluated using three types of event logs: (i) real-life logs, i.e., logs of real-life process execution data; (ii) synthetic logs, generated by replaying real-life process models; and (iii) artificial logs, generated by replaying artificial models.

We found that the majority of methods (31 out of 35) were tested using real-life logs. Among them, 11 approaches (cf. [21], [27], [28], [35], [69], [70], [71], [77], [79], [89], [95]) were further tested against synthetic logs, while 13 approaches (cf. [14], [18], [21], [57], [58], [65], [70], [73], [74], [81], [85], [86], [98]) against artificial logs. Finally, one method was tested both on synthetic and artificial logs only (cf. [75]), while [26], [90] were tested on artificial logs and [51] on synthetic logs only. Among the methods that employ real-life logs, we observed a growing trend in employing publicly-available logs, as opposed to private logs, which hamper the replicability of the results.

Concerning the application domains of the real-life logs, we noticed that several methods used a selection of the logs made available by the Business Process Intelligence Challenge (BPIC), which is held annually as part of the BPM Conference series. These logs are publicly available at the *4TU Centre for Research Data*,[4] and cover domains such as healthcare (used by [21], [35], [40], [48], [73], [87], [94]), banking (used by [21], [35], [40], [64], [69], [74], [79], [83], [87], [94], [98]), IT support management in automotive (cf. [21], [77], [79], [94]), and public administration (cf. [18], [28], [40], [57], [98]). A public log pertaining to a process for managing road traffic fines (also available at the *4TU Centre for Research Data*) was used in [21], [94]. In [86], the authors use logs from various domains available at http://www.processmining.be/actitrac/.

Besides these publicly-available logs, a range of private logs were also used, originating from different domains such as logistics (cf. [89], [91]), traffic congestion dynamics [71], employers habits (cf. [58], [83]), automotive [14], healthcare [21], [74], [94], and project management and insurance (cf. [49], [81]).

4 Benchmark

Using a selection of the methods surveyed in this paper, we conducted an extensive benchmark to identify relative

1. http://promtools.org
2. http://apromore.org
3. http://workcraft.org
4. https://data.4tu.nl/repository/collection:event_logs_real

advantages and tradeoffs. In this section, we justify the criteria of the methods selection, describe the datasets, the evaluation setup and metrics, and present the results of the benchmark. These results, consolidated with the findings from the systematic literature review, are then discussed in Section 5.

4.1 Methods Selection

Assessing all the methods that resulted from the search would not be possible due to the heterogeneous nature of the inputs required and the outputs produced. Hence, we decided to focus on the largest subset of comparable methods. The methods considered were the ones satisfying the following criteria:

i an implementation of the method is publicly accessible;
ii the output of the method is a BPMN model or a Petri net.

Techniques that produce BPMN models were retained because it is a de facto and de jure standard for process modeling, while techniques that produce Petri nets were also included due to the fact that a large number of existing process discovery techniques produce Petri nets as discussed above.

The application of these criteria resulted in an initial selection of the following methods (corresponding to one third of the total studies): α\$ [14], Inductive Miner [41], Evolutionary Tree Miner [18], Fodina [95], Structured Heuristic Miner 6.0 [21], Split Miner [94], Hybrid ILP Miner [105], RegPFA [79], Stage Miner [97], BPMN Miner [81], Decomposed Process Mining [102].

A posteriori, we excluded the latter four due to the following reasons: Decomposed Process Mining, BPMN Miner, and Stage Miner were excluded as such approaches follow a divide-and-conquer approach that could be applied on top of any discovery method to improve its results; on the other hand, we excluded RegPFA because its output is a graphical representation of a Petri net (DOT), which could not be seamlessly serialized into the standard Petri net format.

We also considered including commercial process mining tools in the benchmark. Specifically, we investigated Disco,[5] Celonis,[6] Minit,[7] and myInvenio.[8] Disco and Minit are not able to produce business process models from event logs. Instead, they can only produce directly-follows graphs, which do not have an execution semantics. Indeed, when a given node (task) has several incoming arcs, a directly-follows graph does not tell us whether or not the task in question should wait for all its incoming tasks to complete, or just for one of them, or a subset of them. A similar remark applies to split points in the directly-follows graph. Given their lack of execution semantics, it is not possible to directly translate a directly-follows graph into a BPMN models or a Petri net. Instead, we have to determine the intended behavior at each split and join point, which is precisely what several of the automated process discovery techniques based on directly-follows graph do (e.g., the Inductive Miner or Split Miner).

Celonis and myInvenio can produce BPMN process models but all they do is to insert OR (inclusive) gateways at the split and join points of the process map. To the best of our knowledge, there is no existing technique for measuring precision and fitness (three key measures used for evaluating automated process discovery methods) for BPMN models with OR-joins. When the model does not contain OR-joins, or when the OR-joins are arranged in block-structured topologies, it is possible to translate the BPMN models to Petri nets using existing mappings from BPMN to Petri nets [110]. Once the model is translated as a Petri net, it becomes possible to use existing techniques for assessing fitness and precision available for Petri nets. But when OR-joins appear in arbitrary topologies, including unstructured cycles, this approach cannot be applied.

In conclusion, the final selection of methods for the benchmark contained: α\$, Inductive Miner (IM), Evolutionary Tree Miner (ETM), Fodina (FO), Structured Heuristic Miner 6.0 (S-HM$_6$), Split Miner (SM), and Hybrid ILP Miner (HILP).

4.2 Evaluation Metrics

For all the selected discovery methods, we measured the following accuracy and complexity metrics: recall (a.k.a. fitness), precision, generalization, complexity, and soundness.

Fitness measures the ability of a model to reproduce the behavior contained in a log. Under trace semantics, a fitness of 1 means that the model can reproduce every trace in the log. In this paper, we use the fitness measure proposed in [111], which measures the degree to which every trace in the log can be aligned (with a small number of errors) with a trace produced by the model. In other words, this measure tells us how close traces in the log are to traces that can be generated by the model on average.

Precision measures the ability of a model to generate only the behavior found in the log. A score of 1 indicates that any trace produced by the model is contained in the log. In this paper, we use the precision measure defined in [112], which is based on similar principles as the above fitness measure. Recall and precision can be combined into a single measure known as F-score, which is the harmonic mean of the two measurements $\left(2 \cdot \frac{Fitness \cdot Precision}{Fitness + Precision}\right)$.

Generalization refers to the ability of an automated discovery algorithm to discover process models that generate traces that are not present in the log but that can be produced by the business process under observation. In other words, an automated process discovery algorithm has a high generalization on a given event log if it is able to discover a process model from the event log, which generates traces that: (i) are not in the event log; but (ii) can be produced by the business process that generated the event log. Note that unlike fitness and precision, generalization is a property of an algorithm on an event log, and not a property of the model produced by an algorithm when applied to a given event log.

In line with the above definition, we use k-fold cross-validation [113] to measure generalization. This k-fold cross-validation approach to measure generalization has been advocated in several studies in the field of automated process discovery [2], [114], [115], [116]. Concretely, we divide the log into k parts, we discover a model from $k-1$ parts (i.e., we hold-out one part), and measure the fitness of the discovered model against the part held out. This is repeated for every possible part held out. Generalization is the mean of the fitness values obtained for each part held out. A generalization of one means that the discovered model produces traces in the observed process, even if those traces are not in the log from which the model was discovered. The algorithm we implemented to measure the

5. http://fluxicon.com/disco
6. http://www.celonis.com
7. http://minitlabs.com/
8. http://www.my-invenio.com

generalization computes the folds of each log randomly, i.e., starting from the complete log, a random set of traces is selected to compose each fold. To compare the generalization of different discovered methods, we ensured that the folds given as input to each discovery method were always the same. In the results reported below, we use a value of $k = 3$ for performance reasons (as opposed to the classical value of $k = 10$). The fitness calculation for most of the algorithm-log pairs is slow, and repeating it 10 times for every algorithm-log combination is costly. To test if the results could be affected by this choice of k, we used $k = 10$ for SM and IM on the BPIC12 log, and found that the value of the 10-fold generalization measure was within one percentage point of that of the 3-fold generalization measure.

Complexity quantifies how difficult it is to understand a model. Several complexity metrics have been shown to be (inversely) related to understandability [117], including *Size* (number of nodes); *Control-Flow Complexity (CFC)* (the amount of branching caused by gateways in the model) and *Structuredness* (the percentage of nodes located directly inside a block-structured single-entry single-exit fragment). In the context of this work, we will consider each metric independently since there is no established approach to combine these metrics in a single complexity metric.

In the following, we provide the individual formulas used to compute these three metrics:

- Size: $S_N(G) = |N|$, where N is the set of nodes of the process model G;
- CFC: $CFC(G) = \sum_{c \in S_{and}} 1 + \sum_{c \in S_{xor}} |c_{xor}\bullet| + \sum_{c \in S_{or}} 2^{|c_{or}\bullet|} - 1$, where S_{and}, S_{xor}, and S_{or} are the sets of AND, XOR, and OR connectors of the process model G and $\bullet$ identifies the nodes directly reachable from a connector;
- Structuredness: $\Phi = 1 - \frac{S_N(G')}{S_N(G)}$, where G is the original process model and G' is the reduced process model.

Lastly, *soundness* assesses the behavioral quality of a process model by reporting whether the model violates one of the three soundness criteria [118]: (i) option to complete; (ii) proper completion; and (iii) no dead transitions.

4.3 Setup and Datasets

To guarantee the reproducibility of our benchmark and to provide the community with a tool for comparing new methods with the ones evaluated in this paper, we developed a command-line Java application that performs measurements of accuracy and complexity metrics on the discovery methods selected above, against all the logs used in our benchmark. The only exception was ETM, which we could not embed in our tool due to its complex configuration settings, hence we relied on its ProM implementation. The tool can be easily extended to incorporate new logs. Moreover, it is possible to include additional discovery methods and metrics by implementing two predefined interfaces. This is possible through the use of *Java reflection*, which allows the tool to automatically detect the presence of new algorithms and metrics. More information about the interfaces to implement, how to include them in the benchmark, and the benchmark source code structure are available in the Readme files provided with the tool.[9]

For our evaluation, we used two datasets. The first is the collection of real-life event logs publicly available at the *4TU Centre for Research Data* as of March 2017.[10] Out of this collection, we considered the *BPI Challenge* (BPIC) logs, the *Road Traffic Fines Management Process* (RTFMP) log, and the *SEPSIS Cases* log. These logs record executions of business processes from a variety of domains, e.g., healthcare, finance, government and IT service management. For our evaluation, we held out those logs that do not explicitly capture business processes (i.e., the BPIC 2011 and 2016 logs), and those contained in other logs (e.g., the *Environmental permit application process* log). Finally, in seven logs (i.e., the BPIC14, BPIC15 collection, and BPIC17 logs), we applied the filtering technique proposed in [119] to remove infrequent behavior.[11] This filtering step was necessary since all the models discovered by the considered methods exhibited very poor accuracy (F-score close to 0 or not computable) on the above logs, making the comparison useless.

Table 3 reports the characteristics of the twelve logs used. These logs are widely heterogeneous ranging from simple to very complex, with a log size ranging from 681 traces (for the $\text{BPIC15}_{2\text{f}}$ log) to 150,370 traces (for the RTFMP log). A Similar variety can be observed in the percentage of distinct traces, ranging from 0.2 to 80.6 percent, and the number of event classes (i.e., activities executed within the process), ranging from 7 to 82. Finally, the length of a trace also varies from very short, with traces containing only one event, to very long with traces containing 185 events.

The second dataset is composed of twelve proprietary logs sourced from several companies around the world. Table 4 reports the characteristics of these logs. Also in this case, the logs are quite heterogeneous, with the number of traces (and the percentage of distinct traces) ranging from 225 (of which 99.9 percent distinct) to 787,657 (of which 0.01 percent distinct). The number of recorded events varies between 4,434 and 2,099,835, whilst the number of event classes ranges from 8 to 310.

We performed two types of evaluations. In the first evaluation, we compared all the process discovery methods using their default parameters. In the second one, we analyzed to what extent each discovery method could improve its output using hyper-parameter optimization. Due to the extremely-long execution times, it was prohibitive to hyper-parameter optimize the $\alpha\$$ and ETM methods. So we held out these two methods from the second evaluation. Additionally, we excluded HILP since we did not find any input parameters that could be used to optimize the F-score of the models produced. For the remaining four methods, we evaluated the following input parameters: the two filtering thresholds required as input by SM and S-HM_6, the single threshold required as input by IM, and the threshold and the boolean flag required as input by FO. All the thresholds ranged from 0.0 to 1.0. Specifically, to appreciate

9. The tool and its source code are available at https://doi.org/10.5281/zenodo.1219321. For the latest version of the source code, refer to https://github.com/raffaeleconforti/ResearchCode.

10. https://data.4tu.nl/repository/collection:event_logs_real

11. This technique uses a parameter called "percentile", which refers to the percentile of the distribution of the frequency of the arcs in the directly-follows graph extracted from the log, to automatically determine the frequency threshold for the filtering. We set this parameter to its default value of 12.5 percent.

TABLE 3
Descriptive Statistics of Public Logs

Log Name	Total traces	Dist. traces (%)	Total events	Dist. events	Tr. length min	Tr. length avg	Tr. length max
BPIC12	13,087	33.4	262,200	36	3	20	175
$BPIC13_{cp}$	1,487	12.3	6,660	7	1	4	35
$BPIC13_{inc}$	7,554	20.0	65,533	13	1	9	123
$BPIC14_{f}$	41,353	36.1	369,485	9	3	9	167
$BPIC15_{1f}$	902	32.7	21,656	70	5	24	50
$BPIC15_{2f}$	681	61.7	24,678	82	4	36	63
$BPIC15_{3f}$	1,369	60.3	43,786	62	4	32	54
$BPIC15_{4f}$	860	52.4	29,403	65	5	34	54
$BPIC15_{5f}$	975	45.7	30,030	74	4	31	61
$BPIC17_{f}$	21,861	40.1	714,198	41	11	33	113
RTFMP	150,370	0.2	561,470	11	2	4	20
SEPSIS	1,050	80.6	15,214	16	3	14	185

TABLE 4
Descriptive Statistics of Proprietary Logs

Log Name	Total traces	Dist. traces (%)	Total events	Dist. events	Tr. length min	Tr. length avg	Tr. length max
PRT1	12,720	8.1	75,353	9	2	5	64
PRT2	1,182	97.5	46,282	9	12	39	276
PRT3	1,600	19.9	13,720	15	6	8	9
PRT4	20,000	29.7	166,282	11	6	8	36
PRT5	739	0.01	4,434	6	6	6	6
PRT6	744	22.4	6,011	9	7	8	21
PRT7	2,000	6.4	16,353	13	8	8	11
PRT8	225	99.9	9,086	55	2	40	350
PRT9	787,657	0.01	1,808,706	8	1	2	58
PRT10	43,514	0.01	78,864	19	1	1	15
PRT11	174,842	3.0	2,099,835	310	2	12	804
PRT12	37,345	7.5	163,224	20	1	4	27

variance in the discovered models, we used steps of 0.05 for IM, steps of 0.10 for the thresholds of SM and FO, and steps of 0.20 for $S\text{-}HM_6$. For FO, we considered all the possible combinations of the filtering threshold and the boolean flag, while, for $S\text{-}HM_6$, we used steps of 0.20. In terms of logs, in the second evaluation, we considered all logs except PRT11, because all methods failed to generate a model from this log (except ETM), as evidenced by the results of the first evaluation with default parameters.

We performed the first evaluation on a 6-core Intel Xeon CPU E5-1650 v3 @ 3.50 GHz with 128 GB RAM running Java 8. We allocated a total of 16 GB to the heap space and 10 GB to the stack space. We enforced a timeout of four hours for the discovery phase and one hour for measuring each of the quality metrics. We ran the second evaluation on a 6-core Intel Xeon CPU E5-2699 v4 @ 2.20 GHz with 128 GB RAM running Java 8, and we increased the heap and stack spaces to 25 GB and 15 GB respectively, using a timeout of 24 hours for each method-log evaluation.

4.4 Benchmark Results

The results of the default parameters evaluation are shown in Tables 5, 6, 7, and 8. In the tables, we used "-" to report that a given accuracy or complexity measurement could not be reliably obtained due to syntactical or behavioral issues in the discovered model (i.e., a disconnected model or an unsound model). Additionally, to report the occurrence of a timeout or an exception during the execution of a discovery method, we used "*t/o*" and "*ex*", respectively. We highlighted in bold the best score for each measure on each log, we underlined the second-best score, and we summarized these achievements in Table 9.

The first evaluation shows the absence of a clear winner among the discovery methods tested, although almost each of them clearly showed specific benefits and drawbacks.

HILP experienced severe difficulties in producing useful outputs. The method often produced disconnected models or models containing multiple end places without providing information about the final marking (a well defined final marking is required in order to measure fitness and precision). Due to these difficulties, we could only assess model complexity for HILP, except for the simplest event log (the PRT5), where HILP had performance comparable to the other methods.

$\alpha\$$ showed scalability issues, timing out in eight event logs (33 percent of the times). Although none of the discovered models stood out in accuracy or in complexity, $\alpha\$$ in general produced models striking a good balance between fitness and precision (except for the $BPIC13_{inc}$ log).

FO struggled to deliver sound models, discovering only eight sound models. Nevertheless, its outputs were usually highly fitting, scoring the best fitness four times out of 24.

$S\text{-}HM_6$ performed better than FO, although its discovered models were also often unsound. Out of the 16 sound models discovered, nine scored the best fitness and generalization, making $S\text{-}HM_6$ the best discovery method for these two quality dimensions along with IM (see Table 9). Precision varied based on the input event log, demonstrating that the performance of $S\text{-}HM_6$ is bounded to the type of input log. Whilst it was not always the best in F-score, $S\text{-}HM_6$ achieved good results, scoring often the second-best F-score.

The remaining three methods, namely IM, ETM, and SM, consistently performed very well across the whole evaluation, excelling either in fitness, precision, F-score or generalization, and simultaneously striking the highest simplicity for the discovered process models. IM scored 20 times a fitness greater than 0.90 (of which 9 times the highest), and it achieved similar results for generalization. Despite this, IM did not stand out for its precision, nor for its F-score. ETM and SM achieved respectively 19 and 21 times a precision greater than 0.80, and ETM's precision was the best 11 times. However, ETM scored high precision at the cost of lower fitness. Lastly, SM stood out for its F-score (i.e., high and balanced fitness and precision), achieving an F-score above 0.80 20 times out of 24, outperforming the other methods 17 times. Despite these remarkable results, SM does not guarantee soundness by design. As a result, it produced one unsound model out of 24 (from the PRT11 log).

In terms of complexity, IM, ETM, and SM stood out among all methods (see Table 9). IM and ETM always discovered sound and fully block-structured models (structuredness equal to 1.00). ETM and SM discovered the smallest or second-smallest model for more than 50 percent of the logs. These models also had low CFC, and were the ones with the lowest CFC for 13 logs (ETM) and for four logs (SM). For the execution time, SM was the clear winner. It systematically outperformed all the other methods, regardless of the input. It was the fastest discovery method 23 times out of 24, discovering a model in less than a second for 19 logs. In contrast, ETM was the slowest method, reaching the timeout of four hours for 22 logs.

The results of the hyper-parameter optimization evaluation are shown in Tables 10, 11, 12, and 13. Here we marked

TABLE 5
Default Parameters Evaluation Results for the BPIC Logs

Log	Discovery Method	Accuracy			Gen.	Complexity			Sound	Exec. Time(s)
		Fitness	Precision	F-score	(3-Fold)	Size	CFC	Struct.		
BPIC12	α$	t/o	t/o	t/o	t/o	t/o	t/o	t/o	t/o	t/o
	IM	**0.98**	0.50	0.66	**0.98**	**59**	37	**1.00**	yes	6.60
	ETM	0.44	**0.82**	0.57	t/o	67	**16**	**1.00**	yes	14,400
	FO	-	-	-	-	102	117	0.13	no	9.66
	S-HM$_6$	-	-	-	-	88	46	0.40	no	227.80
	HILP	-	-	-	-	300	460	-	no	772.20
	SM	0.97	0.72	**0.83**	0.97	63	43	0.73	yes	**0.58**
BPIC13$_{cp}$	α$	-	-	-	-	18	9	-	no	10,112.60
	IM	0.82	**1.00**	0.90	0.82	**9**	4	**1.00**	yes	0.10
	ETM	**1.00**	0.70	0.82	t/o	38	38	**1.00**	yes	14,400
	FO	-	-	-	-	25	23	0.60	no	0.06
	S-HM$_6$	0.94	0.99	**0.97**	0.94	15	6	**1.00**	yes	130.0
	HILP	-	-	-	-	10	**3**	-	yes	0.10
	SM	0.99	0.93	0.96	**0.99**	13	7	**1.00**	yes	**0.03**
BPIC13$_{inc}$	α$	0.35	0.91	0.51	t/o	15	7	0.47	yes	4,243.14
	IM	0.92	0.54	0.68	0.92	13	7	**1.00**	yes	1.00
	ETM	**1.00**	0.51	0.68	t/o	32	144	**1.00**	yes	14,400
	FO	-	-	-	-	43	54	0.77	no	1.41
	S-HM$_6$	0.91	**0.96**	0.93	0.91	**9**	**4**	**1.00**	yes	0.80
	HILP	-	-	-	-	24	9	-	yes	2.50
	SM	0.98	0.92	**0.95**	**0.98**	15	10	**1.00**	yes	**0.23**
BPIC14$_f$	α$	0.47	0.63	0.54	t/o	62	36	0.31	yes	14,057.48
	IM	**0.89**	0.64	0.74	**0.89**	31	18	**1.00**	yes	3.40
	ETM	0.61	**1.00**	0.76	t/o	**23**	**9**	**1.00**	yes	14,400
	FO	-	-	-	-	37	46	0.38	no	27.73
	S-HM$_6$	-	-	-	-	202	132	0.73	no	147.40
	HILP	-	-	-	-	80	59	-	no	7.30
	SM	0.77	0.91	**0.84**	0.78	24	15	**1.00**	yes	**0.59**
BPIC15$_{1f}$	α$	0.71	0.76	0.73	t/o	219	91	0.22	yes	3,545.9
	IM	0.97	0.57	0.71	**0.96**	164	108	**1.00**	yes	0.60
	ETM	0.56	**0.94**	0.70	t/o	**67**	**19**	**1.00**	yes	14,400
	FO	**1.00**	0.76	0.87	0.94	146	91	0.25	yes	1.02
	S-HM$_6$	-	-	-	-	204	116	0.56	no	128.10
	HILP	-	-	-	-	282	322	-	no	4.40
	SM	0.90	0.88	**0.89**	0.89	114	43	0.48	yes	**0.48**
BPIC15$_{2f}$	α$	-	-	-	-	348	164	0.08	no	8,787.48
	IM	0.93	0.56	0.70	0.94	193	123	**1.00**	yes	0.70
	ETM	0.62	**0.91**	0.74	t/o	**95**	**32**	**1.00**	yes	14,400
	FO	-	-	-	-	195	159	0.09	no	0.61
	S-HM$_6$	**0.98**	0.59	0.74	**0.97**	259	150	0.29	yes	163.2
	HILP	-	-	-	-	-	-	-	-	t/o
	SM	0.77	0.90	**0.83**	0.75	124	41	0.32	yes	**0.25**
BPIC15$_{3f}$	α$	-	-	-	-	319	169	0.03	no	10,118.15
	IM	**0.95**	0.55	0.70	**0.95**	159	108	**1.00**	yes	1.30
	ETM	0.68	0.88	0.76	t/o	**84**	**29**	**1.00**	yes	14,400
	FO	-	-	-	-	174	164	0.06	no	0.89
	S-HM$_6$	**0.95**	0.67	0.79	**0.95**	159	151	0.13	yes	139.90
	HILP	-	-	-	-	433	829	-	no	1,062.90
	SM	0.78	**0.94**	**0.85**	0.78	92	**29**	0.61	yes	**0.36**
BPIC15$_{4f}$	α$	-	-	-	-	272	128	0.13	no	6,410.25
	IM	0.96	0.58	0.73	0.96	162	111	**1.00**	yes	0.7
	ETM	0.65	**0.93**	0.77	t/o	**83**	**28**	**1.00**	yes	14,400
	FO	-	-	-	-	157	127	0.14	no	0.50
	S-HM$_6$	**0.99**	0.64	0.78	**0.99**	209	137	0.37	yes	136.90
	HILP	-	-	-	-	364	593	-	no	14.7
	SM	0.73	0.91	**0.81**	0.74	98	31	0.31	yes	**0.25**
BPIC15$_{5f}$	α$	0.62	0.75	0.68	t/o	280	126	0.10	yes	7,603.19
	IM	0.94	0.18	0.30	0.94	134	95	**1.00**	yes	1.50
	ETM	0.57	**0.94**	0.71	t/o	**88**	**18**	**1.00**	yes	14,400
	FO	**1.00**	0.71	0.83	**1.00**	166	125	0.15	yes	0.56
	S-HM$_6$	**1.00**	0.70	0.82	**1.00**	211	135	0.35	yes	141.90
	HILP	-	-	-	-	-	-	-	-	t/o
	SM	0.79	**0.94**	**0.86**	0.78	105	30	0.33	yes	**0.27**
BPIC17$_f$	α$	t/o	t/o	t/o	t/o	t/o	t/o	t/o	t/o	t/o
	IM	**0.98**	0.70	0.82	**0.98**	**35**	20	**1.00**	yes	13.30
	ETM	0.76	**1.00**	0.86	t/o	42	**4**	**1.00**	yes	14,400
	FO	-	-	-	-	98	82	0.25	no	64.33
	S-HM$_6$	0.95	0.62	0.75	0.94	42	13	0.97	yes	143.20
	HILP	-	-	-	-	222	330	-	no	384.50
	SM	0.96	0.81	**0.88**	0.96	39	21	**1.00**	yes	**2.53**

TABLE 6
Default Parameters Evaluation Results for the Public Logs

Log	Discovery Method	Accuracy			Gen.	Complexity			Sound?	Exec. Time (sec)
		Fitness	Precision	F-score	(3-Fold)	Size	CFC	Struct.		
RTFMP	α$	t/o	t/o	t/o	t/o	t/o	t/o	t/o	t/o	t/o
	IM	0.99	0.70	0.82	0.99	34	20	**1.00**	yes	10.90
	ETM	0.99	0.92	0.95	t/o	57	32	**1.00**	yes	14,400
	FO	**1.00**	0.94	0.97	0.97	31	32	0.19	yes	2.57
	S-HM$_6$	0.98	0.95	0.96	0.98	163	97	**1.00**	yes	262.70
	HILP	-	-	-	-	57	53	-	no	3.50
	SM	**1.00**	**0.97**	**1.00**	**1.00**	**25**	**18**	0.40	yes	**1.25**
SEPSIS	α$	-	-	-	-	146	156	0.01	no	3,883.12
	IM	**0.99**	0.45	0.62	**0.96**	50	32	**1.00**	yes	0.40
	ETM	0.83	0.66	0.74	t/o	108	101	**1.00**	yes	14,400
	FO	-	-	-	-	60	63	0.28	no	0.17
	S-HM$_6$	0.92	0.42	0.58	0.92	279	198	**1.00**	yes	242.70
	HILP	-	-	-	-	87	129	-	no	1.60
	SM	0.76	**0.77**	**0.77**	0.77	**39**	**25**	0.82	yes	**0.05**

TABLE 7
Default Parameters Evaluation Results for the Proprietary Logs - Part 1/2

Log	Discovery Method	Accuracy			Gen.	Complexity			Sound	Exec. Time(s)
		Fitness	Precision	F-score	(3-Fold)	Size	CFC	Struct.		
PRT1	α$	-	-	-	t/o	45	34	-	no	11,168.54
	IM	0.90	0.67	0.77	0.90	**20**	**9**	**1.00**	yes	2.08
	ETM	**0.99**	0.81	0.89	t/o	23	12	**1.00**	yes	14,400
	FO	-	-	-	-	30	28	0.53	no	0.95
	S-HM$_6$	0.88	0.77	0.82	0.88	59	39	**1.00**	yes	122.16
	HILP	-	-	-	-	195	271	-	no	2.59
	SM	0.98	**0.99**	**0.98**	**0.98**	27	16	**1.00**	yes	**0.47**
PRT2	α$	-	-	-	-	134	113	0.25	no	3,438.72
	IM	ex	ex	ex	ex	45	33	**1.00**	yes	1.41
	ETM	0.57	**0.94**	0.71	t/o	86	**21**	**1.00**	yes	14,400
	FO	-	-	-	-	76	74	0.59	no	0.88
	S-HM$_6$	-	-	-	-	67	105	0.43	no	1.77
	HILP	-	-	-	-	190	299	-	no	21.33
	SM	**0.81**	0.70	**0.75**	**0.81**	**38**	28	0.87	yes	**0.31**
PRT3	α$	0.67	0.76	0.71	0.67	70	40	0.11	yes	220.11
	IM	0.98	0.68	0.80	0.98	37	20	**1.00**	yes	0.44
	ETM	0.98	0.86	**0.92**	t/o	51	37	**1.00**	yes	14,400
	FO	**1.00**	0.86	**0.92**	**1.00**	34	37	0.32	yes	0.50
	S-HM$_6$	**1.00**	0.83	0.91	**1.00**	40	38	0.43	yes	0.67
	HILP	-	-	-	-	343	525	-	no	0.73
	SM	0.82	**0.92**	0.87	0.84	**29**	**13**	0.76	yes	**0.17**
PRT4	α$	0.86	0.93	0.90	t/o	**21**	**10**	**1.00**	yes	13,586.48
	IM	0.93	0.75	0.83	0.93	27	13	**1.00**	yes	1.33
	ETM	0.84	0.85	0.84	t/o	64	28	**1.00**	yes	14,400
	FO	-	-	-	-	37	40	0.54	no	6.33
	S-HM$_6$	**1.00**	0.86	**0.93**	**1.00**	370	274	**1.00**	yes	241.57
	HILP	-	-	-	-	213	306	-	no	5.31
	SM	0.83	**1.00**	0.91	0.88	31	19	0.77	yes	**0.45**
PRT5	α$	1.00	1.00	1.00	1.00	10	1	1.00	yes	2.02
	IM	1.00	1.00	1.00	1.00	10	1	1.00	yes	0.03
	ETM	1.00	1.00	1.00	1.00	10	1	1.00	yes	2.49
	FO	1.00	1.00	1.00	1.00	10	1	1.00	yes	**0.02**
	S-HM$_6$	1.00	1.00	1.00	1.00	10	1	1.00	yes	0.11
	HILP	1.00	1.00	1.00	1.00	10	1	1.00	yes	0.05
	SM	1.00	1.00	1.00	1.00	10	1	1.00	yes	**0.02**
PRT6	α$	0.80	0.77	0.79	0.80	38	17	0.24	yes	40.10
	IM	0.99	0.82	0.90	0.99	23	10	**1.00**	yes	2.30
	ETM	0.98	0.80	0.88	t/o	41	16	**1.00**	yes	14,400
	FO	**1.00**	0.91	0.95	**1.00**	22	17	0.41	yes	0.05
	S-HM$_6$	**1.00**	0.91	0.95	**1.00**	22	17	0.41	yes	0.42
	HILP	-	-	-	-	157	214	-	no	0.13
	SM	0.94	**1.00**	**0.97**	0.94	**15**	**4**	**1.00**	yes	**0.02**

TABLE 8
Default Parameters Evaluation Results for the Proprietary Logs - Part 2/2

Log	Discovery Method	Accuracy			Gen.	Complexity			Sound?	Exec. Time (sec)
		Fitness	Precision	F-score	(3-Fold)	Size	CFC	Struct.		
PRT7	α\$	0.85	0.90	0.88	0.85	29	**9**	0.48	yes	143.66
	IM	**1.00**	0.73	0.84	**1.00**	29	13	**1.00**	yes	0.13
	ETM	0.90	0.81	0.85	t/o	60	29	**1.00**	yes	14,400
	FO	0.99	**1.00**	0.99	0.99	**26**	16	0.39	yes	0.08
	S-HM$_6$	**1.00**	**1.00**	**1.00**	**1.00**	163	76	**1.00**	yes	249.74
	HILP	-	-	-	-	278	355	-	no	0.27
	SM	0.91	**1.00**	0.95	0.92	29	10	0.48	yes	**0.06**
PRT8	α\$	t/o	t/o	t/o	t/o	t/o	t/o	t/o	t/o	t/o
	IM	**0.98**	0.33	0.49	**0.93**	111	92	**1.00**	yes	**0.41**
	ETM	0.35	**0.88**	0.50	t/o	**75**	**12**	**1.00**	yes	14,400
	FO	-	-	-	-	228	179	0.74	no	0.55
	S-HM$_6$	-	-	-	-	388	323	0.87	no	370.66
	HILP	t/o	t/o	t/o	t/o	t/o	t/o	t/o	t/o	t/o
	SM	0.97	0.41	**0.57**	0.93	241	322	0.82	yes	1.28
PRT9	α\$	t/o	t/o	t/o	t/o	t/o	t/o	t/o	t/o	t/o
	IM	0.90	0.61	0.73	0.89	28	16	**1.00**	yes	63.70
	ETM	0.75	0.49	0.59	0.74	**27**	**13**	**1.00**	yes	1,266.71
	FO	-	-	-	-	32	45	0.72	no	42.83
	S-HM$_6$	**0.96**	0.98	**0.97**	**0.96**	723	558	**1.00**	yes	318.69
	HILP	-	-	-	-	164	257	-	no	51.47
	SM	0.92	**1.00**	0.96	0.92	29	19	**1.00**	yes	**9.11**
PRT10	α\$	t/o	t/o	t/o	t/o	t/o	t/o	t/o	t/o	t/o
	IM	0.96	0.79	0.87	0.96	**41**	**29**	**1.00**	yes	2.50
	ETM	**1.00**	0.63	0.77	t/o	61	45	**1.00**	yes	14,400
	FO	0.99	0.93	**0.96**	**0.99**	52	85	0.64	yes	0.98
	S-HM$_6$	-	-	-	-	77	110	-	no	1.81
	HILP	-	-	-	-	846	3130	-	no	2.55
	SM	0.97	**0.95**	**0.96**	0.97	60	49	0.75	yes	**0.47**
PRT11	α\$	t/o	t/o	t/o	t/o	t/o	t/o	t/o	t/o	t/o
	IM	t/o	t/o	t/o	t/o	549	365	**1.00**	yes	121.50
	ETM	**0.10**	**1.00**	**0.18**	t/o	**21**	**3**	**1.00**	yes	14,400
	FO	-	-	-	-	680	713	0.68	no	81.33
	S-HM$_6$	ex	ex	ex	ex	ex	ex	ex	ex	ex
	HILP	t/o	t/o	t/o	t/o	t/o	t/o	t/o	t/o	t/o
	SM	-	-	-	-	712	609	0.12	no	**19.53**
PRT12	α\$	t/o	t/o	t/o	t/o	t/o	t/o	t/o	t/o	t/o
	IM	**1.00**	0.77	0.87	**1.00**	32	25	**1.00**	yes	3.94
	ETM	0.63	**1.00**	0.77	t/o	**21**	**8**	**1.00**	yes	14,400
	FO	-	-	-	-	87	129	0.38	no	1.67
	S-HM$_6$	-	-	-	-	4370	3191	**1.00**	yes	347.57
	HILP	-	-	-	-	926	2492	-	no	7.34
	SM	0.96	0.97	**0.97**	0.96	78	65	0.78	yes	**0.36**

with a "*" the discovery methods that were not able to complete the exploration of the solution space within 24 hours of timeout time. The purpose of this second evaluation was to understand if the discovery methods can achieve higher F-score when optimally tuned, and what price they pay for such an improvement, i.e., at the cost of a detriment of which other quality dimensions. In line with our goal, Tables 10 and 11 report the accuracy and complexity scores of the discovered models with the highest F-score. We note that some of the insights gained from the default parameters evaluation do not hold anymore. FO and S-HM$_6$ were almost always able to discover sound models from each log for at least one input configuration. FO outperformed IM in fitness, by scoring the best fitness 14 times. IM performed better in precision (being eight times the best), yet falling behind SM, which was the most accurate in precision (12 times out of 23). Further, IM delivered the simplest model for size, CFC and structuredness all at once, 14 times, followed only by SM (six times). Despite S-HM$_6$ was only in less than one third of the logs the best in any accuracy dimension and never the best for size and CFC, its performance was very close to the best for the majority of the times. SM confirmed to be the most balanced discovery method, scoring the highest F-score most of the times (18 out of 23), at the same time discovering very simple models along with IM. The results for generalization indicate that SM scored the best generalization eight times, followed very closely by FO (seven times) and by S-HM$_6$ (six times), with IM achieving the highest generalization in four cases only.

Finally, Tables 12 and 13 report the best score that each discovery method can achieve in each dimension.[12] FO and IM are always able to maximally optimize fitness, scoring always 1.00 as their best score. S-HM$_6$ and SM perform slightly worse in fitness, the former being always

12. Full results of the hyper-parameter optimization evaluation are included in the zip file available at https://doi.org/10.5281/zenodo.1219321

TABLE 9
Best Score Frequencies for Each Quality Dimension (Default Parameters Evaluation)

Log	Discovery Method	Accuracy			Gen.	Complexity			Exec.
		Fitness	Precision	F-score	(3-Fold)	Size	CFC	Struct.	Time (sec)
Frequency Absolute Best	α$	0	0	0	0	1	2	0	0
	IM	**9**	1	0	**9**	5	2	**24**	1
	ETM	5	**11**	2	0	**10**	**13**	**24**	0
	FO	4	1	2	4	1	0	0	0
	S-HM$_6$	**9**	2	4	**9**	1	1	8	0
	HILP	0	0	0	0	0	1	0	0
	SM	2	10	**17**	5	4	4	7	**23**
Frequency Second Best	α$	0	3	0	0	1	1	1	0
	IM	8	2	3	**9**	**9**	**13**	0	6
	ETM	3	4	**7**	0	1	2	0	0
	FO	2	2	5	2	4	0	6	**17**
	S-HM$_6$	1	4	6	2	1	1	7	1
	HILP	0	0	0	0	1	0	0	0
	SM	**10**	**9**	3	8	**9**	7	**11**	0
Total	α$	0	3	0	0	2	3	1	0
	IM	**17**	3	3	**18**	**14**	**15**	**24**	7
	ETM	8	15	9	0	11	**15**	**24**	0
	FO	6	3	7	6	5	0	6	17
	S-HM$_6$	10	6	10	11	2	2	15	1
	HILP	0	0	0	0	1	1	0	0
	SM	12	**19**	**20**	13	13	11	18	**23**

in the range 0.90-1.00 and the latter in the range 0.80-1.00. Similar are the results for generalization, with FO and IM leading again, and S-HM$_6$ and SM following suit closely. As for precision, FO and S-HM$_6$ get over 0.80 66 percent of the times only, whilst SM and IM strike better results, being always in the range 0.90-1.00 (excluding the two outlier models from the logs PRT2 and PRT8). Finally, F-score results reveal the ability of the discovery methods to properly balance fitness and precision. We note that IM lacks such an ability, reaching an F-score above 0.90 only 30 percent of the times, with values below 0.80 35 percent of the times. These results are as such despite IM can reach very high values both in fitness and in precision, individually. FO follows IM as worst performer in F-score with similar outcomes, whilst S-HM$_6$ and SM distinguished themselves with scores in the range 0.80-1.00 and often over 0.90 (more than 50 percent of the times). As for model simplicity, IM leads the way, followed by SM. Whilst FO and S-HM$_6$ struggle to optimize both size and CFC, most of the times falling behind IM and SM.

In conclusion, a method outperforming all others across all metrics could not be identified. Despite this, when it comes to default parameters IM, ETM and SM showed to be the most effective methods when the focus is either on fitness, precision, or F-score, and these methods all yield simple process models. However, even these three methods suffer from a common weakness, which is their inability to handle large-scale real-life logs, as reported for the PRT11 log in our evaluation. On the other hand, the hyper-parameter optimization exercise showed that also FO and S-HM$_6$ can perform very well, though at the expenses of long execution times (up to 24 hours for some logs) and powerful computational resources.

5 Discussion

Our review highlights a growing interest in the field of automated process discovery, and confirms the existence of a wide and heterogeneous number of proposals. Despite such a variety, we can clearly identify two main streams: methods that output procedural process models, and methods that output declarative process models. Further, procedural approaches provide various language alternatives, though, most of these methods output Petri nets.

The predominance of Petri nets is driven by the expressive power of this language, and by the requirements of the methods used to assess the quality of the discovered process models (chiefly, fitness and precision). Despite some modeling languages have a straightforward conversion to Petri nets, the strict requirements of these quality assessment tools represent a limitation for the proposals in this research field. For the same reason, it was not possible to compare the two main streams, so we decided to focus our evaluation and comparison on the procedural methods, which have been more investigated in the literature than their declarative counterparts and are used more in practical scenarios.

Our benchmark shows the benefits of procedural automated process discovery methods, as well as their limitations. These latter include lack of scalability for large and complex logs, and strong differences in the output models, across the various quality metrics. Regarding this aspect, the majority of methods were not able to excel in accuracy or complexity, except for IM, ETM and SM. Indeed, these three methods were the only ones to consistently perform very well in fitness (IM), precision (ETM, SM), F-score (SM), complexity (IM, ETM, SM) and execution time (SM). Nevertheless, our evaluation shows that even IM, ETM and SM can fail when challenged with large-scale unfiltered real-life events logs, as shown in the case of the PRT11 log.

To conclude, even if many proposals are available in this research area, and some of them are able to systematically deliver good to optimal results, there is still space for research and improvements. Furthermore, it is important to highlight that the great majority of the methods do not have a working or available implementation. This hampers their systematic evaluation, so we can only rely on the results

TABLE 10
Scores of the Models with the Best F-Score Discovered with Hyper-Parameter Optimization (Public Logs)

Log	Discovery Method	Accuracy			Gen. (3-Fold)	Complexity		
		Fitness	Precision	F-score		Size	CFC	Struct.
BPIC12	IM	0.90	0.69	0.78	0.91	69	46	**1.00**
	FO*	**1.00**	0.07	0.14	-	112	1369	**1.00**
	S-HM$_6$	0.96	0.67	0.79	0.96	97	110	0.63
	SM	0.97	**0.72**	**0.83**	**0.97**	**63**	**43**	0.73
BPIC13$_{cp}$	IM	**0.99**	**0.98**	**0.98**	**0.99**	**11**	**5**	**1.00**
	FO	0.00	0.42	0.00	-	19	16	**1.00**
	S-HM$_6$	0.96	0.92	0.94	0.96	20	14	0.80
	SM	**0.99**	0.93	0.96	**0.99**	13	7	**1.00**
BPIC13$_{inc}$	IM	0.90	0.87	0.89	0.90	**13**	**5**	**1.00**
	FO	0.00	0.36	0.00	-	42	76	0.88
	S-HM$_6$	0.93	**0.98**	**0.96**	0.93	16	10	**1.00**
	SM	**0.98**	0.92	0.95	**0.98**	15	10	**1.00**
BPIC14$_f$	IM	0.75	**0.97**	0.85	0.75	**19**	**4**	**1.00**
	FO	**0.97**	0.81	**0.88**	0.31	27	34	0.56
	S-HM$_6$	0.91	0.84	**0.88**	**0.91**	178	117	0.97
	SM	0.85	0.86	0.86	0.85	30	22	0.70
BPIC15$_{1f}$	IM	0.81	0.68	0.74	0.83	140	70	**1.00**
	FO	**1.00**	0.76	0.87	0.94	146	91	0.26
	S-HM$_6$	0.88	**0.89**	0.89	0.58	1576	550	**1.00**
	SM	0.95	0.86	**0.90**	**0.95**	**122**	**51**	0.45
BPIC15$_{2f}$	IM	0.71	0.76	0.74	0.69	**141**	61	**1.00**
	FO	**0.99**	0.63	0.77	**0.99**	195	164	0.09
	S-HM$_6$	**0.99**	0.62	0.76	**0.99**	246	167	0.19
	SM	0.81	**0.86**	**0.83**	0.81	**141**	**58**	0.31
BPIC15$_{3f}$	IM	0.65	**0.99**	0.79	0.63	**73**	**8**	**1.00**
	FO	**0.99**	0.60	0.75	**0.99**	162	163	0.07
	S-HM$_6$	0.81	0.77	0.79	0.81	231	77	0.97
	SM	0.78	0.94	**0.85**	0.78	92	29	0.61
BPIC15$_{4f}$	IM	0.73	0.84	0.78	0.75	108	42	**1.00**
	FO	**1.00**	0.67	0.80	**1.00**	155	128	0.14
	S-HM$_6$	0.99	0.66	0.79	0.99	217	145	0.36
	SM	0.77	**0.90**	**0.83**	0.78	**102**	**35**	0.34
BPIC15$_{5f}$	IM	0.64	0.88	0.74	0.65	**105**	**34**	**1.00**
	FO	**1.00**	0.71	0.83	**1.00**	166	125	0.15
	S-HM$_6$	0.82	**0.94**	0.87	0.81	610	166	0.96
	SM	0.84	0.92	**0.88**	0.82	108	36	0.22
BPIC17$_f$	IM	**1.00**	0.70	0.82	**1.00**	39	24	**1.00**
	FO*	-	-	-	-	-	-	-
	S-HM$_6$	0.97	0.70	0.81	0.97	51	25	**1.00**
	SM	0.94	**0.83**	**0.88**	0.94	**37**	**19**	**1.00**
RTFMP	IM	0.94	0.98	0.96	0.94	28	**10**	**1.00**
	FO	**1.00**	0.94	0.97	0.84	31	32	0.19
	S-HM$_6$	0.95	**0.99**	0.97	0.95	82	30	**1.00**
	SM	**1.00**	0.97	**0.98**	**1.00**	**25**	18	0.40
SEPSIS	IM	0.62	**0.98**	0.76	0.76	**31**	**14**	**1.00**
	FO	**0.96**	0.36	0.53	0.30	51	109	0.33
	S-HM$_6$	0.80	0.39	0.52	**0.86**	299	187	**1.00**
	SM	0.76	0.77	**0.77**	0.77	39	25	0.82

reported in the respective papers. Finally, for those methods we assessed, we were not able to identify a unique winner, since the best methods showed to either maximize fitness, precision or F-score. Despite these considerations, it can be noted that there has been significant progress in this field in the past five years. Indeed, IM, ETM and SM outperformed the discovery methods developed in the previous decade, as well as their extensions (i.e., AGNEs Miner and S-HM$_6$).

6 Threats to Validity

The first threat to validity refers to the potential selection bias and inaccuracies in data extraction and analysis typical of literature reviews. In order to minimize such issues, our systematic literature review carefully adheres to the guidelines outlined in [4]. Concretely, we used well-known literature sources and libraries in information technology to extract relevant works on the topic of automated process

TABLE 11
Scores of the Models with the Best F-Score Discovered with Hyper-Parameter Optimization (Proprietary Logs)

Log	Discovery Method	Accuracy			Gen.	Complexity		
		Fitness	Precision	F-score	(3-Fold)	Size	CFC	Struct.
PRT1	IM	0.91	0.89	0.90	0.91	**24**	**11**	**1.00**
	FO	**0.98**	0.92	0.95	**0.99**	25	29	0.72
	S-HM$_6$	0.95	0.97	0.96	0.95	37	29	0.92
	SM	**0.98**	**0.98**	**0.98**	0.98	27	16	**1.00**
PRT2	IM	-	-	-	-	-	-	-
	FO	**1.00**	0.17	0.30	**1.00**	55	241	**0.93**
	S-HM$_6$	-	-	-	-	-	-	-
	SM	0.81	**0.70**	**0.75**	0.81	**38**	**28**	0.87
PRT3	IM	0.87	**0.93**	0.90	0.87	**27**	**8**	**1.00**
	FO	**1.00**	0.86	**0.92**	**1.00**	34	37	0.32
	S-HM$_6$	0.99	0.85	0.91	0.96	40	34	0.48
	SM	0.95	0.89	**0.92**	0.95	33	24	0.55
PRT4	IM	0.86	**1.00**	0.92	0.86	**21**	**5**	**1.00**
	FO	**1.00**	0.87	0.93	-	32	41	0.50
	S-HM$_6$	0.93	0.96	**0.95**	0.93	66	55	0.77
	SM	0.97	0.93	**0.95**	**0.97**	36	32	0.56
PRT5	IM	1.00	1.00	1.00	**1.00**	12	1	**1.00**
	FO	1.00	1.00	1.00	0.95	**10**	1	**1.00**
	S-HM$_6$	1.00	1.00	1.00	**1.00**	12	1	**1.00**
	SM	1.00	1.00	1.00	**1.00**	**10**	1	**1.00**
PRT6	IM	0.90	**1.00**	0.95	0.90	17	**2**	**1.00**
	FO	**1.00**	0.91	0.95	0.96	22	17	0.41
	S-HM$_6$	0.98	0.96	**0.97**	**0.98**	24	15	0.46
	SM	0.94	**1.00**	**0.97**	0.94	**15**	4	**1.00**
PRT7	IM	0.88	**1.00**	0.93	0.88	**23**	**5**	**1.00**
	FO	0.99	**1.00**	0.99	0.99	26	16	0.39
	S-HM$_6$	**1.00**	**1.00**	**1.00**	**1.00**	165	76	**1.00**
	SM	0.93	**1.00**	0.96	0.92	34	16	0.29
PRT8	IM*	**1.00**	0.09	0.16	**0.99**	**95**	**86**	**1.00**
	FO	-	-	-	-	-	-	-
	S-HM$_6$	0.93	0.42	0.58	0.89	221	422	0.83
	SM	0.77	**0.58**	**0.66**	0.76	214	176	0.93
PRT9	IM	0.93	0.71	0.80	0.93	**28**	**14**	**1.00**
	FO	-	-	-	-	-	-	-
	S-HM$_6$	**0.99**	0.99	**0.99**	**0.99**	41	59	0.68
	SM	**0.99**	**1.00**	**0.99**	**0.99**	41	34	0.68
PRT10	IM	**1.00**	0.81	0.89	**1.00**	**47**	**33**	**1.00**
	FO	0.99	0.93	0.96	-	52	85	0.64
	S-HM$_6$	0.98	0.83	0.90	0.98	1440	972	**1.00**
	SM	0.98	**0.95**	**0.97**	0.98	64	55	0.66
PRT12	IM	0.93	0.92	0.93	0.93	**37**	**26**	**1.00**
	FO	**1.00**	0.80	0.89	0.94	60	237	0.87
	S-HM$_6$	0.88	0.67	0.76	0.88	3943	2314	**1.00**
	SM	0.97	**0.97**	**0.97**	**0.97**	80	75	0.76

discovery. Further, we performed a backward reference search to avoid the exclusion of potentially relevant papers. Finally, to avoid that our review was threatened by insufficient reliability, we ensured that the search process could be replicated by other researchers. However, the search may produce different results as the algorithm used by source libraries to rank results based on relevance may be updated (see, e.g., Google Scholar).

The experimental evaluation on the other hand is limited in scope to techniques that produce Petri nets (or models in languages such as BPMN or Process Trees, which can be directly translated to Petri nets). Also, it only considers main studies identified in the SLR with an available implementation. In order to compensate for these shortcomings, we published the benchmarking toolset as open-source software in order to enable researchers both to reproduce the results herein reported and to run the same evaluation for other methods, or for alternative configurations of the evaluated methods.

Another limitation is the use of only 24 event logs, which to some extent limits the generalizability of the conclusions. However, the event logs included in the evaluation are all real-life logs of different sizes and features, including different application domains. In addition, to mitigate this limitation, we have structured the released benchmarking toolset in such a way that the benchmark can be seamlessly rerun with additional datasets.

Finally, we selected the folds of each log randomly when computing the 3-fold generalization. We checked the fitness values obtained for the three folds of each algorithm, and found variations in these values ranging from 1 to 3

TABLE 12
Best Scores Achieved in Hyper-Parameter Evaluation by Each Approach on Each Quality Dimension (Public Logs)

Metric	Discovery Method	BPIC Logs 12	13_{cp}	13_{inc}	14_f	15_{1f}	15_{2f}	15_{3f}	15_{4f}	15_{5f}	17_f	RTFMP	SEPSIS
Fitness	IM	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
	FO	**1.00**	0.00	0.00	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	-	**1.00**	**1.00**
	S-HM_6	0.96	**1.00**	0.93	**1.00**	**1.00**	0.99	0.99	**1.00**	**1.00**	**1.00**	0.95	0.80
	SM	0.98	**1.00**	**1.00**	0.85	0.97	0.95	0.81	0.83	0.92	0.96	**1.00**	0.98
Prec.	IM	**0.92**	**1.00**	0.89	**1.00**	**0.97**	**1.00**	**0.99**	0.91	**0.99**	**0.89**	0.98	**0.98**
	FO	0.07	0.42	0.36	0.81	0.76	0.63	0.60	0.67	0.71	-	0.94	0.36
	S-HM_6	0.67	0.92	**0.98**	0.84	0.89	0.78	0.77	0.66	0.94	0.70	0.99	0.39
	SM	0.80	0.93	0.92	0.91	0.92	0.92	0.94	**0.93**	0.96	0.83	**1.00**	0.81
F-score	IM	0.78	**0.98**	0.89	0.85	0.74	0.74	0.79	0.78	0.74	0.82	0.96	0.76
	FO	0.14	0.00	0.00	**0.88**	0.87	0.77	0.75	0.80	0.83	-	0.97	0.53
	S-HM_6	0.79	0.94	**0.96**	**0.88**	0.89	0.76	0.79	0.79	0.87	0.81	0.97	0.52
	SM	**0.83**	0.96	0.95	0.86	**0.90**	**0.83**	**0.85**	**0.83**	**0.88**	**0.88**	**0.98**	**0.77**
Gen. (3-Fold)	IM	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
	FO	-	-	0.00	0.96	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	-	0.90	0.94
	S-HM_6	0.96	0.99	0.93	**1.00**	**1.00**	0.99	0.99	**1.00**	**1.00**	**1.00**	0.95	0.86
	SM	0.98	**1.00**	**1.00**	0.85	0.97	0.94	0.82	0.85	0.91	0.96	**1.00**	0.98
Size	IM	**15**	9	**9**	17	**63**	**30**	**17**	**25**	**32**	23	**11**	**14**
	FO	112	19	39	27	113	137	114	111	117	-	23	40
	S-HM_6	87	**8**	16	**13**	74	246	207	139	232	**22**	82	299
	SM	53	13	15	23	107	117	92	92	103	36	22	36
CFC	IM	**8**	2	**3**	2	10	**7**	**8**	**9**	**9**	6	**5**	**9**
	FO	1369	16	54	34	47	57	53	46	44	-	13	35
	S-HM_6	65	**0**	10	**0**	**0**	167	77	35	140	**0**	30	187
	SM	28	7	10	15	35	34	28	27	28	19	10	21
Struct.	IM	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
	FO	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.29	**1.00**	**1.00**	**1.00**	-	**1.00**	**1.00**
	S-HM_6	0.77	**1.00**	**1.00**	0.97	**1.00**	0.99	0.98	0.37	0.96	**1.00**	**1.00**	**1.00**
	SM	**1.00**	**1.00**	**1.00**	**1.00**	0.48	0.41	0.62	0.35	0.33	**1.00**	0.54	0.92

percentage points, suggesting that the algorithms produced different results for different folds. However, given that the folds were generated randomly, we have no guarantee that there is sufficient difference between the folds.

7 Related Work

A previous survey and benchmark of automated process discovery methods has been reported by De Weerdt et al. [3]. This survey covered 27 approaches, and it assessed seven of them. We used it as starting point for our study.

The benchmark reported by [3] includes seven approaches, namely AGNEsMiner, $\alpha+$, $\alpha++$, Genetic Miner (and a variant thereof), Heuristics Miner and ILP Miner. In comparison, our benchmark includes $\alpha\$$ (which is an improved version of $\alpha+$ and $\alpha++$), Structured Heuristics miner (which is an extension of Heuristics Miner), Hybrid ILP Miner (an improvement of ILP), Evolutionary Tree Miner (which is a genetic algorithm postdating the evaluation of [3]) Notably, we did not include AGNEsMiner due to the very long execution times (as suggested by the authors in a conversation over emails exchanged during this work).

Another difference with respect to the previous survey is that in our paper we based our evaluation both on public and proprietary event logs, whilst the evaluation of [3] is solely based on artificial event logs and closed datasets, due to the unavailability of public datasets at the time of that study.

In terms of results, in [3], the authors found that Heuristic Miner achieved a better F-score than other approaches and generally produced simpler models, while ILP achieved the best fitness at the expense of low precision and high model complexity. Our results show that SM achieves even better F-score and lower model complexity than other techniques, followed by ETM and IM, which excelled for precision and fitness (respectively). Thus, it appears that in the last years progress has been pursued successfully.

Another previous survey in the field is outdated [120] and a more recent one is not intended to be comprehensive [121], but rather limits on plug-ins available in the ProM toolset. Another related effort is CoBeFra—a tool suite for measuring fitness, precision and model complexity of automatically discovered process models [122].

8 Conclusion

This article presented a Systematic Literature Review (SLR) of automated process discovery methods and a comparative evaluation of existing implementations of these methods using a benchmark covering twelve publicly-available real-life event logs, twelve proprietary real-life event logs, and nine quality metrics. The toolset used in this benchmark is available as open-source software and 50 percent of the event logs are publicly available. The benchmarking toolset has been designed in a way that it can be seamlessly extended with additional methods, event logs, and evaluation metrics.

The SLR put into evidence a vast number of automated process discovery methods (344 relevant papers were analyzed). Traditionally, many of these proposals produce

TABLE 13
Best Scores Achieved in Hyper-Parameter Evaluation by Each Approach on Each Quality Dimension (Proprietary Logs)

Metric	Discovery Method	Proprietary (PRT) Logs										
		# 1	# 2	# 3	# 4	# 5	# 6	# 7	# 8	# 9	# 10	# 12
Fitness	IM	**1.00**	-	**1.00**	**1.00**	1.00	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
	FO	**1.00**	**1.00**	**1.00**	**1.00**	1.00	**1.00**	**1.00**	-	-	**1.00**	**1.00**
	S-HM$_6$	**1.00**	-	**1.00**	**1.00**	1.00	**1.00**	**1.00**	**1.00**	0.99	**1.00**	0.88
	SM	**1.00**	0.83	**1.00**	0.97	1.00	0.94	0.93	0.99	0.99	**1.00**	**1.00**
Prec.	IM	0.89	-	**0.93**	**1.00**	1.00	**1.00**	**1.00**	0.09	0.89	0.95	**0.99**
	FO	0.92	0.17	0.89	0.94	1.00	0.91	**1.00**	-	-	**0.96**	0.80
	S-HM$_6$	0.97	-	0.85	0.96	1.00	0.96	**1.00**	0.42	0.99	0.83	0.67
	SM	**0.98**	**0.71**	0.92	**1.00**	1.00	**1.00**	**1.00**	**0.58**	**1.00**	0.95	0.98
F-score	IM	0.90	-	0.90	0.92	1.00	0.95	0.93	0.16	0.80	0.89	0.93
	FO	0.95	0.30	**0.92**	0.93	1.00	0.95	0.99	-	-	0.96	0.89
	S-HM$_6$	0.96	-	0.91	**0.95**	1.00	**0.97**	**1.00**	0.58	0.99	0.90	0.76
	SM	**0.98**	**0.75**	**0.92**	**0.95**	1.00	**0.97**	0.96	**0.66**	**0.99**	**0.97**	**0.97**
Gen. (3-Fold)	IM	**1.00**	-	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**0.99**	**1.00**	**1.00**	**1.00**
	FO	**1.00**	**1.00**	**1.00**	-	0.95	0.96	**1.00**	-	-	-	0.94
	S-HM$_6$	**1.00**	-	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.98	0.99	**1.00**	0.88
	SM	**1.00**	0.83	**1.00**	0.97	**1.00**	0.94	0.92	0.94	0.99	**1.00**	0.99
Size	IM	14	-	**27**	**21**	12	17	23	95	**16**	27	**25**
	FO	22	55	30	30	10	22	**21**	-	-	48	60
	S-HM$_6$	**13**	-	37	66	12	24	40	**59**	41	**23**	3943
	SM	27	**33**	29	29	**8**	**13**	**21**	210	29	52	66
CFC	IM	4	-	**8**	**5**	1	**2**	**5**	86	**2**	10	**18**
	FO	16	74	24	33	1	17	7	-	-	44	66
	S-HM$_6$	**0**	-	27	55	1	15	15	**0**	58	**0**	2314
	SM	16	**24**	13	18	**0**	3	6	176	19	39	49
Struct.	IM	**1.00**	-	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
	FO	0.77	0.93	0.57	0.54	**1.00**	0.41	0.81	-	-	0.81	0.87
	S-HM$_6$	**1.00**	-	0.72	**1.00**	**1.00**	0.50	**1.00**	0.85	**1.00**	**1.00**	**1.00**
	SM	**1.00**	**1.00**	0.76	0.77	**1.00**	**1.00**	0.76	0.93	**1.00**	0.89	0.90

Petri nets, but more recently, there has been an increasing number of methods that produce models in other languages, including BPMN and declarative constraints. We also noticed a recent emphasis on producing block-structured process models.

The results of the empirical evaluation show that methods that seek to produce block-structured process models (Inductive Miner and Evolutionary Tree Miner) achieve the best performance in terms of fitness or precision, and complexity. On the other hand, methods that do not restrict the topology of the generated process models (Split Miner) produce process models of higher quality in terms of F-score, although these methods cannot guarantee soundness (though they can guarantee deadlock-freedom). We also observed that in the case of very complex event logs, it is necessary to use a filtering method prior to applying existing automated process discovery methods. Without this filtering, the precision of the resulting models is close to zero. A direction for future work is to develop automated process discovery techniques that incorporate adaptive filtering approaches so that they can auto-tune themselves to deal with very complex logs.

Another limitation observed while conducting the benchmark was the lack of universal measures of fitness and precision, which would be applicable not only to Petri nets (or BPMN models that can be mapped to Petri nets), but equally well to declarative or data-driven process modeling notations. Developing more universal measures of fitness and precision is another possible target of future work.

Acknowledgments

This research is partly funded by the Australian Research Council (grant DP150103356) and the Estonian Research Council (grant IUT20-55). It is also partly supported by the H2020-RISE EU project FIRST (734599), the Sapienza grant DAKIP and the Italian projects Social Museum and Smart Tourism (CTN01_00034_23154), NEPTIS (PON03PE_00214_3), and RoMA - Resilience of Metropolitan Areas (SCN_00064).

References

[1] W. M. P. van der Aalst, A. J. M. M. Weijters, and L. Maruster, "Workflow mining: Discovering process models from event logs," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 9, pp. 1128–1142, Sep. 2004.

[2] W. M. P. van der Aalst, *Process Mining: Data Science in Action*. Berlin, Germany: Springer, 2016.

[3] J. De Weerdt, M. De Backer, J. Vanthienen, and B. Baesens, "A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs," *Inf. Syst.*, vol. 37, no. 7, pp. 654–676, 2012.

[4] B. Kitchenham, "Procedures for performing systematic reviews," Keele University, Keele, U.K., Tech. Rep. TR/SE-0401, 2004.

[5] W. M. P. van der Aalst, A. Kalenkova, V. Rubin, and H. M. W. Verbeek, "Process discovery using localized events," in *Proc. Int. Conf. Appl. Theory Petri Nets Concurrency*, 2015, pp. 287–308.

[6] F. Folino, M. Guarascio, and L. Pontieri, "On the discovery of explainable and accurate behavioral models for complex lowly-structured business processes," in *Proc. 17th Int. Conf. Enterprise Inf. Syst.*, 2015, pp. 206–217.

[7] A. Fink, *Conducting Research Literature Reviews: From the Internet to Paper*, 3rd ed. Newbury Park, CA, USA: Sage, 2010.

[8] C. Okoli and K. Schabram, "A guide to conducting a systematic literature review of information systems research," *Sprouts: Work. Papers Inf. Syst.*, vol. 10, no. 26, pp. 1–49, 2010.
[9] J. Randolph, "A guide to writing the dissertation literature review," *Practical Assessment Res. Eval.*, vol. 14, no. 13, pp. 1–13, 2009.
[10] R. Torraco, "Writing integrative literature reviews: Guidelines and examples," *Human Resource Develop. Rev.*, vol. 4, no. 3, pp. 356–367, 2005.
[11] W. M. P. van der Aalst, A. J. M. M. Weijters, and L. Maruster, "Workflow mining: Discovering process models from event logs," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 9, pp. 1128–1142, Sep. 2004.
[12] A. K. Alves de Medeiros, B. F. van Dongen, W. M. P. van der Aalst, and A. J. M. M. Weijters, "Process mining: Extending the α-algorithm to mine short loops," BETA Working Paper Series, vol. 113, pp. 145–180, 2004.
[13] L. Wen, W. M. P. van der Aalst, J. Wang, and J. Sun, "Mining process models with non-free-choice constructs," *Data Mining Knowl. Discovery*, vol. 15, no. 2, pp. 145–180, 2007.
[14] Q. Guo, L. Wen, J. Wang, Z. Yan, and S. Y. Philip, "Mining invisible tasks in non-free-choice constructs," in *Proc. Int. Conf. Bus. Process Manage.*, 2015, pp. 109–125.
[15] S. Goedertier, D. Martens, J. Vanthienen, and B. Baesens, "Robust process discovery with artificial negative events," *J. Mach. Learn. Res.*, vol. 10, no. Jun, pp. 1305–1340, 2009.
[16] A. K. Alves de Medeiros and A. J. M. M. Weijters, "Genetic process mining," in *Proc. Int. Conf. Appl. Theory Petri Nets*, 2005, pp. 48–69.
[17] A. K. Alves de Medeiros, A. J. M. M. Weijters, and W. M. P. van der Aalst, "Genetic process mining: An experimental evaluation," *Data Mining Knowl. Discovery*, vol. 14, no. 2, pp. 245–304, 2007.
[18] J. C. A. M. Buijs, B. F. van Dongen, and W. M. P. van der Aalst, "Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity," *Int. J. Cooperative Inf. Syst.*, vol. 23, no. 1, 2014, Art. no. 1440001.
[19] A. J. M. M. Weijters and W. M. P. van der Aalst, "Rediscovering workflow models from event-based data using little thumb," *Integr. Comput.-Aided Eng.*, vol. 10, no. 2, pp. 151–162, 2003.
[20] A. J. M. M. Weijters, W. M. P. van der Aalst, and A. K. Alves de Medeiros, "Process mining with the heuristics miner-algorithm," Technische Universiteit Eindhoven, Tech. Rep. WP, vol. 166, pp. 1–34, 2006.
[21] A. Augusto, R. Conforti, M. Dumas, and M. La Rosa, "Automated discovery of structured process models from event logs: The discover-and-structure approach," *Data Knowl. Eng.*, 2018, doi: https://doi.org/10.1016/j.datak.2018.04.007.
[22] F. Mannhardt, M. de Leoni, H. A. Reijers, and W. M. P. van der Aalst, "Data-driven process discovery-revealing conditional infrequent behavior from event logs," in *Proc. Int. Conf. Adv. Inf. Syst. Eng.*, 2017, pp. 545–560.
[23] A. J. M. M. Weijters and J. Ribeiro, "Flexible heuristics miner (FHM)," in *Proc. IEEE Symp. Comput. Intell. Data Mining*, 2011, pp. 310–317.
[24] S. De Cnudde, J. Claes, and G. Poels, "Improving the quality of the Heuristics Miner in ProM 6.2," *Expert Syst. Appl.*, vol. 41, no. 17, pp. 7678–7690, 2014.
[25] J. M. E. van der Werf, B. F. van Dongen, C. A. Hurkens, and A. Serebrenik, "Process discovery using Integer Linear Programming," *Fundam. Inf.*, vol. 94, no. 3–4, pp. 387–412, 2009.
[26] S. J. van Zelst, B. F. van Dongen, W. M. P. van der Aalst, and H. M. W. Verbeek, "Discovering Workflow nets using Integer Linear Programming," *Comput.*, pp. 1–28, 2017.
[27] Z. Huang and A. Kumar, "A study of quality and accuracy tradeoffs in process mining," *INFORMS J. Comput.*, vol. 24, no. 2, pp. 311–327, 2012.
[28] F. M. Maggi, R. P. J. C. Bose, and W. M. P. van der Aalst, "Efficient discovery of understandable declarative process models from event logs," in *Proc. 24th Int. Conf. Adv. Inf. Syst. Eng.*, 2012, pp. 270–285.
[29] F. M. Maggi, A. J. Mooij, and W. M. P. van der Aalst, "User-guided discovery of declarative process models," in *Proc. IEEE Symp. Comput. Intell. Data Mining*, 2011, pp. 192–199.
[30] F. M. Maggi, R. P. J. C. Bose, and W. M. P. van der Aalst, "A knowledge-based integrated approach for discovering and repairing declare maps," in *Proc. Int. Conf. Adv. Inf. Syst. Eng.*, 2013, pp. 433–448.
[31] M. L. Bernardi, M. Cimitile, and F. M. Maggi, "Discovering cross-organizational business rules from the cloud," in *Proc. IEEE Symp. Comput. Intell. Data Mining*, 2014, pp. 389–396.
[32] F. M. Maggi, "Discovering metric temporal business constraints from event logs," in *Proc. 13th Int. Conf. Perspectives Bus. Inf. Res.*, 2014, pp. 261–275.
[33] T. Kala, F. M. Maggi, C. Di Ciccio, and C. Di Francescomarino, "Apriori and sequence analysis for discovering declarative process models," in *Proc. IEEE 20th Int. Enterprise Distrib. Object Comput. Conf.*, 2016, pp. 1–9.
[34] F. M. Maggi, C. Di Ciccio, C. Di Francescomarino, and T. Kala, "Parallel algorithms for the automated discovery of declarative process models," *Inf. Syst.*, vol. 74, pp. 136–152, 2018.
[35] C. Di Ciccio and M. Mecella, "A two-step fast algorithm for the automated discovery of declarative workflows," in *Proc. IEEE Symp. Comput. Intell. Data Mining*, 2013, pp. 135–142.
[36] C. Di Ciccio and M. Mecella, "Mining constraints for artful processes," in *Proc. 15th Int. Conf. Bus. Inf. Syst.*, 2012, pp. 11–23.
[37] C. Di Ciccio and M. Mecella, "On the discovery of declarative control flows for artful processes," *ACM Trans. Manage. Inf. Syst.*, vol. 5, no. 4, pp. 24:1–24:37, 2015.
[38] C. Di Ciccio, F. M. Maggi, and J. Mendling, "Discovering target-branched declare constraints," in *Proc. Int. Conf. Bus. Process Manage.*, 2014, pp. 34–50.
[39] C. Di Ciccio, F. M. Maggi, and J. Mendling, "Efficient discovery of target-branched declare constraints," *Inf. Syst.*, vol. 56, pp. 258–283, 2016.
[40] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Discovering block-structured process models from event logs containing infrequent behaviour," in *Proc. Int. Bus. Process Manage. Workshops*, 2013, pp. 66–78.
[41] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Discovering block-structured process models from event logs-A constructive approach," in *Proc. 34th Int. Conf. Appl. Theory Petri Nets Concurrency*, 2013, pp. 311–329.
[42] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Discovering block-structured process models from incomplete event logs," in *Proc. 35th Int. Conf. Appl. Theory Petri Nets Concurrency*, 2014, pp. 91–110.
[43] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Exploring processes and deviations," in *Proc. Int. Workshops Bus. Process Manage.*, 2014, pp. 304–316.
[44] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Using life cycle information in process discovery," in *Proc. 13th Int. Workshops Bus. Process Manage.*, 2015, pp. 204–217.
[45] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Scalable process discovery with guarantees," in *Proc. Int. Conf. Enterprise, Bus.-Process Inf. Syst. Model.*, 2015, pp. 85–101.
[46] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Scalable process discovery and conformance checking," *Softw. Syst. Model.*, vol. 17, no. 2, pp. 599–631, 2018.
[47] S. J. J. Leemans and W. M. P. van der Aalst, "Modeling and discovering cancelation behavior," in *Proc. OTM Confederated Int. Conf. Move Meaningful Internet Syst.*, 2017, pp. 93–113.
[48] F. M. Maggi, M. Dumas, L. García-Bañuelos, and M. Montali, "Discovering data-aware declarative process models from event logs," in *Business Process Management*. Berlin, Germany: Springer, 2013, pp. 81–96.
[49] M. Abe and M. Kudo, "Business monitoring framework for process discovery with real-life logs," in *Proc. Int. Conf. Bus. Process Manage.*, 2014, pp. 416–423.
[50] M. Kudo, A. Ishida, and N. Sato, "Business process discovery by using process skeletonization," in *Proc. Int. Conf. Signal-Image Technol. Internet-Based Syst.*, 2013, pp. 976–982.
[51] S. K. L. M. vanden Broucke, J. Vanthienen, and B. Baesens, "Declarative process discovery with evolutionary computing," in *IEEE Congress Evol. Comput.*, 2014, pp. 2412–2419.
[52] W. M. P. van der Aalst, J. C. A. M. Buijs, and B. F. van Dongen, "Towards improving the representational bias of process mining," in *Proc. 1st Int. Symp. Data-Driven Process Discovery Anal.*, 2011, pp. 39–54.
[53] J. C. A. M. Buijs, B. F. van Dongen, and W. M. P. van der Aalst, "A genetic algorithm for discovering process trees," in *Proc. IEEE Congress Evol. Comput.*, 2012, pp. 1–8.
[54] J. C. A. M. Buijs, B. F. van Dongen, W. M. P. van der Aalst, "On the role of fitness, precision, generalization and simplicity in process discovery," in *Proc. OTM Conf.*, 2012, pp. 305–322.
[55] J. C. A. M. Buijs, B. F. van Dongen, and W. M. P. van der Aalst, "Discovering and navigating a collection of process models using multiple quality dimensions," in *Proc. Int. Workshops Bus. Process Manage.*, 2013, pp. 3–14.

[56] M. L. van Eck, J. C. A. M. Buijs, and B. F. van Dongen, "Genetic process mining: Alignment-based process model mutation," in *Proc. Int. Workshops Bus. Process Manage.*, 2014, pp. 291–303.
[57] J. Carmona and J. Cortadella, "Process discovery algorithms using numerical abstract domains," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 12, pp. 3064–3076, Dec. 2014.
[58] S. Ferilli, "WoMan: Logic-based workflow learning and management," *IEEE Trans. Syst. Man Cybern.: Syst.*, vol. 44, no. 6, pp. 744–756, Jun. 2014.
[59] S. Ferilli, B. De Carolis, and D. Redavid, "Logic-based incremental process mining in smart environments," in *Proc. Int. Conf. Ind. Eng. Other Appl. Appl. Intell. Syst.*, 2013, pp. 392–401.
[60] B. De Carolis, S. Ferilli, and G. Mallardi, "Learning and recognizing routines and activities in SOFiA," in *Proc. Eur. Conf. Ambient Intell.*, 2014, pp. 191–204.
[61] S. Ferilli, B. De Carolis, and F. Esposito, "Learning complex activity preconditions in process mining," in *Proc. 3rd Int. Workshop New Frontiers Mining Complex Patterns*, 2014, pp. 164–178.
[62] S. Ferilli, D. Redavid, and F. Esposito, "Logic-based incremental process mining," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, 2015, pp. 218–221.
[63] S. Ferilli, "The WoMan formalism for expressing process models," in *Proc. 16th Ind. Conf. Advances Data Mining. Appl. Theoretical Aspects*, 2016, pp. 363–378.
[64] F. M. Maggi, T. Slaats, and H. A. Reijers, "The automated discovery of hybrid processes," in *Proc. Int. Conf. Bus. Process Manage.*, 2014, pp. 392–399.
[65] D. Redlich, T. Molka, W. Gilani, G. S. Blair, and A. Rashid, "Scalable dynamic business process discovery with the constructs competition miner," in *Proc. 4th Int. Symp. Data-Driven Process Discovery Anal.*, 2014, pp. 91–107.
[66] D. Redlich, T. Molka, W. Gilani, G. Blair, and A. Rashid, "Constructs competition miner: Process control-flow discovery of bp-domain constructs," in *Proc. Int. Conf. Bus. Process Manage.*, 2014, pp. 134–150.
[67] D. Redlich, W. Gilani, T. Molka, M. Drobek, A. Rashid, and G. Blair, "Introducing a framework for scalable dynamic process discovery," in *Proc. Enterprise Eng. Work. Conf.*, 2014, pp. 151–166.
[68] D. Redlich, T. Molka, W. Gilani, G. Blair, and A. Rashid, "Dynamic constructs competition miner-occurrence-vs. time-based ageing," in *Proc. Int. Symp. Data-Driven Process Discovery Anal.*, 2014, pp. 79–106.
[69] O. Vasilecas, T. Savickas, and E. Lebedys, "Directed acyclic graph extraction from event logs," in *Proc. Int. Conf. Inf. Softw. Technol.*, 2014, pp. 172–181.
[70] J. De Smedt, J. De Weerdt, and J. Vanthienen, "Fusion miner: Process discovery for mixed-paradigm models," *Decision Support Syst.*, vol. 77, pp. 123–136, 2015.
[71] G. Greco, A. Guzzo, F. Lupia, and L. Pontieri, "Process discovery under precedence constraints," *ACM Trans Knowl. Discovery Data*, vol. 9, no. 4, 2015, Art. no. 32.
[72] G. Greco, A. Guzzo, and L. Pontieri, "Process discovery via precedence constraints," in *Proc. 20th Eur. Conf. Artif. Intell.*, 2012, pp. 366–371.
[73] V. Liesaputra, S. Yongchareon, and S. Chaisiri, "Efficient process model discovery using maximal pattern mining," in *Proc. Int. Conf. Bus. Process Manage.*, 2015, pp. 441–456.
[74] T. Molka, D. Redlich, M. Drobek, X.-J. Zeng, and W. Gilani, "Diversity guided evolutionary mining of hierarchical process models," in *Proc. Annu. Conf. Genetic Evol. Comput.*, 2015, pp. 1247–1254.
[75] B. Vázquez-Barreiros, M. Mucientes, and M. Lama, "ProDiGen: Mining complete, precise and minimal structure process models with a genetic algorithm," *Inf. Sci.*, vol. 294, pp. 315–333, 2015.
[76] B. Vázquez-Barreiros, M. Mucientes, and M. Lama, "A genetic algorithm for process discovery guided by completeness, precision and simplicity," in *Proc. Int. Conf. Bus. Process Manage.*, 2014, pp. 118–133.
[77] M. L. Bernardi, M. Cimitile, C. Di Francescomarino, and F. M. Maggi, "Do activity lifecycles affect the validity of a business rule in a business process?" *Inf. Syst.*, vol. 62. pp. 42–59, 2016.
[78] M. L. Bernardi, M. Cimitile, C. Di Francescomarino, and F. M. Maggi, "Using discriminative rule mining to discover declarative process models with non-atomic activities," in *Proc. 8th Int. Symp. RuleML Rules Web. From Theory Appl. Co-Located 21st Eur. Conf. Artif. Intell.*, 2014, pp. 281–295.
[79] D. Breuker, M. Matzner, P. Delfmann, and J. Becker, "Comprehensible predictive models for business processes," *MIS Quart.*, vol. 40, no. 4, pp. 1009–1034, 2016.
[80] D. Breuker, P. Delfmann, M. Matzner, and J. Becker, "Designing and evaluating an interpretable predictive modeling technique for business processes," in *Proc. Int. Workshops Bus. Process Manage.*, 2014, pp. 541–553.
[81] R. Conforti, M. Dumas, L. García-Bañuelos, and M. La Rosa, "BPMN miner: Automated discovery of BPMN process models with hierarchical structure," *Inf. Syst.*, vol. 56, pp. 284–303, 2016.
[82] R. Conforti, M. Dumas, L. García-Bañuelos, and M. La Rosa, "Beyond tasks and gateways: Discovering BPMN models with subprocesses, boundary events and activity markers," in *Proc. Int. Conf. Bus. Process Manage.*, 2014, pp. 101–117.
[83] M. L. van Eck, N. Sidorova, and W. M. P. van der Aalst, "Discovering and exploring state-based models for multi-perspective processes," in *Proc. Int. Conf. Bus. Process Manage.*, 2016, pp. 142–157.
[84] M. L. van Eck, N. Sidorova, and W. M. P. van der Aalst, "Guided interaction exploration in artifact-centric process models," in *Proc. 19th IEEE Conf. Bus. Inf.*, 2017, pp. 109–118.
[85] C. Li, J. Ge, L. Huang, H. Hu, B. Wu, H. Yang, H. Hu, and B. Luo, "Process mining with token carried data," *Inf. Sci.*, vol. 328, pp. 558–576, 2016.
[86] A. Mokhov, J. Carmona, and J. Beaumont, "Mining conditional partial order graphs from event logs," in *Transactions on Petri Nets and Other Models of Concurrency XI*. Berlin, Germany: Springer, 2016, pp. 114–136.
[87] S. Schönig, A. Rogge-Solti, C. Cabanillas, S. Jablonski, and J. Mendling, "Efficient and customisable declarative process mining with SQL," in *Proc. Int. Conf. Adv. Inf. Syst. Eng.*, 2016, pp. 290–305.
[88] S. Schönig, C. Di Ciccio, F. M. Maggi, and J. Mendling, "Discovery of multi-perspective declarative process models," in *Proc. Int. Conf. Serv.-Oriented Comput.*, 2016, pp. 87–103.
[89] W. Song, H.-A. Jacobsen, C. Ye, and X. Ma, "Process discovery from dependence-complete event logs," *IEEE Trans. Serv. Comput.*, vol. 9, no. 5, pp. 714–727, Sep./Oct. 2016.
[90] T. Tapia-Flores, E. Rodríguez-Pérez, and E. López-Mellado, "Discovering process models from incomplete event logs using conjoint occurrence classes," in *Proc. ATAED@ Petri Nets/ACSD*, 2016, pp. 31–46.
[91] B. N. Yahya, M. Song, H. Bae, S.-O. Sul, and J.-Z. Wu, "Domain-driven actionable process model discovery," *Comput. Industrial Eng.*, vol. 99, no. C, pp. 382–400, 2016.
[92] B. N. Yahya, H. Bae, S.-O. Sul, and J.-Z. Wu, "Process discovery by synthesizing activity proximity and user's domain knowledge," in *Proc. Asia-Pacific Conf. Bus. Process Manage.*, 2013, pp. 92–105.
[93] A. Augusto, R. Conforti, M. Dumas, M. La Rosa, and G. Bruno, "Automated discovery of structured process models: Discover structured versus discover and structure," in *Proc. 35th Int. Conf. Conceptual Model.*, 2016, pp. 313–329.
[94] A. Augusto, R. Conforti, M. Dumas, and M. La Rosa, "Split miner: Discovering accurate and simple business process models from event logs," in *Proc. IEEE Int. Conf. Data Mining*, New Orleans, LA, USA, Nov. 18–21, 2017, pp. 1–10.
[95] S. K. L. M. vanden Broucke and J. De Weerdt, "Fodina: A robust and flexible heuristic process discovery technique," *Decision Support Syst.*, vol. 100, pp. 109–118, 2017.
[96] J. De Weerdt, S. K. L. M. vanden Broucke, and F. Caron, "Bidimensional process discovery for mining BPMN models," in *Proc. Int. Workshops Bus. Process Manage.*, Eindhoven, The Netherlands, Sep. 7–8, 2014, pp. 529–540.
[97] H. Nguyen, M. Dumas, A. H. M. ter Hofstede, M. La Rosa, and F. M. Maggi, "Mining business process stages from event logs," in *Proc. Int. Conf. Adv. Inf. Syst. Eng.*, 2017, pp. 577–594.
[98] H. M. W. Verbeek, W. M. P. van der Aalst, and J. Munoz-Gama, "Divide and conquer: A tool framework for supporting decomposed discovery in process mining," *Comput. J.*, vol. 60, pp. 1–26, 2017.
[99] H. M. W. Verbeek and W. M. P. van der Aalst, "An experimental evaluation of passage-based process discovery," in *Proc. Int. Workshop Bus. Process Intell. Bus. Process Manage.*, vol. 132, pp. 205–210, 2012.
[100] W. M. P. van der Aalst, "Decomposing Petri nets for process mining: A generic approach," *Distrib. Parallel Databases*, vol. 31, no. 4, pp. 471–507, 2013.
[101] B. Hompes, H. M. W. Verbeek, and W. M. P. van der Aalst, "Finding suitable activity clusters for decomposed process discovery," in *Proc. Int. Symp. Data-Driven Process Discovery Anal.*, 2014, pp. 32–57.

[102] H. M. W. Verbeek and W. M. P. van der Aalst, "Decomposed process mining: The ILP case," in *Proc. Int. Workshops Bus. Process Manage.*, Eindhoven, The Netherlands, Sep. 7–8, 2014, pp. 264–276.
[103] W. M. P. van der Aalst and H. M. W. Verbeek, "Process discovery and conformance checking using passages," *Fund. Inf.*, vol. 131, no. 1, pp. 103–138, 2014.
[104] S. J. van Zelst, B. F. van Dongen, and W. M. P. van der Aalst, "Avoiding over-fitting in ILP-based process discovery," in *Int. Conf. Bus. Process Manage.*, 2015, pp. 163–171.
[105] S. J. van Zelst, B. F. van Dongen, and W. M. P. van der Aalst, "ILP-based process discovery using hybrid regions," in *Proc. Int. Workshop Algorithms Theories Anal. Event Data*, 2015, pp. 47–61.
[106] M. Pesic, H. Schonenberg, and W. M. P. van der Aalst, "DECLARE: Full support for loosely-structured processes," in *Proc. 11th IEEE Int. Enterprise Distrib. Object Comput. Conf.*, 2007, pp. 287–300.
[107] M. Westergaard and F. M. Maggi, "Declare: A tool suite for declarative workflow modeling and enactment," in *Proc. Demo Track 9th Conf. Bus. Process Manage.*, vol. 820, pp. 1–5, 2011.
[108] T. Slaats, D. M. M. Schunselaar, F. M. Maggi, and H. A. Reijers, "The semantics of hybrid process models," in *Proc. OTM Confederated Int. Conf. Move Meaningful Internet Syst.*, 2016, pp. 531–551.
[109] M. Westergaard and T. Slaats, "Mixing paradigms for more comprehensible models," in *Proc. Bus. Process Manage.*, 2013, pp. 283–290.
[110] C. Favre, D. Fahland, and H. Völzer, "The relationship between workflow graphs and free-choice workflow nets," *Inf. Syst.*, vol. 47, pp. 197–219, 2015.
[111] A. Adriansyah, B. F. van Dongen, and W. M. P. van der Aalst, "Conformance checking using cost-based fitness analysis," in *Proc. 15th IEEE Int. Enterprise Distrib. Object Comput. Conf.*, 2011, pp. 55–64.
[112] A. Adriansyah, J. Munoz-Gama, J. Carmona, B. F. van Dongen, and W. M. P. van der Aalst, "Alignment based precision checking," in *Proc. BPM Workshops*, vol. 132, pp. 137–149, 2012.
[113] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proc. Int. Joint Conf. Artif. Intell.*, 1995, pp. 1137–1145.
[114] A. Rozinat, A. K. Alves de Medeiros, C. Günther, A. J. M. M. Weijters, and W. M. P. van der Aalst, "Towards an evaluation framework for process mining algorithms," BPM Center Report BPM-07–06, 2007.
[115] A. Bolt, M. de Leoni, and W. M. P. van der Aalst, "Scientific workflows for process mining: Building blocks, scenarios, and implementation," *Softw. Tools Technol. Transfer*, vol. 18, no. 6, pp. 607–628, 2016.
[116] B. F. van Dongen, J. Carmona, T. Chatain, and F. Taymouri, "Aligning modeled and observed behavior: A compromise between computation complexity and quality," in *Proc. 29th Int. Conf. Adv. Inf. Syst. Eng.*, 2017, pp. 94–109.
[117] J. Mendling, *Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness*. Berlin, Germany: Springer, 2008.
[118] W. M. P. van der Aalst, *Verification of Workflow Nets*. Berlin, Germany: Springer, 1997, pp. 407–426.
[119] R. Conforti, M. La Rosa, and A. H. M. ter Hofstede, "Filtering out infrequent behavior from business process event logs," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 2, pp. 300–314, Feb. 2017.
[120] W. M. P. van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. J. M. M. Weijters, "Workflow mining: A survey of issues and approaches," *Data Knowl. Eng.*, vol. 47, no. 2, pp. 237–267, 2003.
[121] J. Claes and G. Poels, "Process Mining and the ProM Framework: An exploratory survey," in *Proc. Bus. Process Manage. Workshops*, 2012, pp. 187–198.
[122] S. K. L. M. vanden Broucke, J. De Weerdt, J. Vanthienen, and B. Baesens, "A comprehensive benchmarking framework (CoBeFra) for conformance analysis between procedural process models and event logs in ProM," in *Proc. IEEE Symp. Comput. Intell. Data Mining*, 2013, pp. 254–261.

Adriano Augusto received the graduate degree in computer engineering from the Polytechnic of Turin, Italy, in 2016, presenting a master thesis in the field of process mining. He is working toward the Joint PhD degree at the University of Tartu, Estonia, and the University of Melbourne, Australia.

Raffaele Conforti is a lecturer with the University of Melbourne, Australia. He conducts research on process mining and automation, with a focus on automated process discovery, quality improvement of process event logs, and process-risk management.

Marlon Dumas is a professor of information systems, University of Tartu, Estonia. His research interests span across the fields of software engineering, information systems, and business process management. He is co-author of the textbook *Fundamentals of Business Process Management* (Springer, 2nd edition).

Marcello La Rosa is a professor of information systems, University of Melbourne, Australia. His research interests include process mining, consolidation, and automation. He leads the Apromore Initiative, a cross-university collaboration for the development of an advanced process analytics platform, and co-authored the textbook *Fundamentals of Business Process Management* (Springer, 2nd edition).

Fabrizio Maria Maggi is an associate professor of information systems at the University of Tartu, Estonia. His research interests span across the fields of business process management, information systems and data science. He has published over 100 research papers and articles on the above topics.

Andrea Marrella is a research fellow at Sapienza Università di Roma. His research interests include human-computer interaction, user experience design, knowledge representation, reasoning about action, automated planning, business process management. He has published over 50 research papers on the above topics. He is currently information director of the ACM Journal of Data and Information Quality.

Massimo Mecella is an associate professor at Sapienza Università di Roma. His research focuses on service oriented computing, business process management, cyber-physical systems and Internet-of-things, advanced interfaces, and human-computer interaction. He published more than 150 research papers and chaired different conferences in the above areas.

Allar Soo is working toward the master's of software engineering degree at the University of Tartu. His masters thesis is focused on automated process discovery and its use in practical settings.