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, 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 covering twelve publicly-available real-life event logs and eight quality metrics. The results highlight gaps and unexplored tradeoffs in the field, including the lack of scalability of several 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 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.

In order to be useful, such automatically discovered process models must accurately reflect the behavior recorded in or implied by the log. Specifically, the process model discovered from an event log should: (i) parse the traces in the log; (ii) parse traces that are not in the log but are likely to belong to the process that produced the log; and (iii) not parse 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 grossly 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 article aims to fill this gap by: (i) providing a systematic review of automated process discovery methods; and (ii) a comparative evaluation of six implementations of representative methods, using an open-source benchmark consisting of twelve publicly-available real-life event logs and eight 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 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 config-

- *A. Augusto, M. Dumas, F.M. Maggi, A. Soo are with the University of Tartu, Estonia.*
  E-mail: {adriano.augusto,marlon.dumas,f.m.maggi}@ut.ee, allar.soo@gmail.com
- *A. Augusto, R. Conforti and M. La Rosa are with Queensland University of Technology, Australia.*
  E-mail: {a.augusto,raffaele.conforti,m.larosa}@qut.edu.au
- *A. Marrella and M. Mecella are with Sapienza University of Rome, Italy.*
  E-mail: {marrella,mecella}@diag.uniroma1.it

Manuscript received XX; revised XX.

uration effort.

The rest of the article is structured as follows. Section 2 describes the search protocol used for the systematic literature review, while Section 3 presents classifies the methods identified in the review. Next, Section 4 introduces the experimental benchmark and results, while 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 in [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 a range of inclusion and exclusion criteria to select the studies retrieved through the search.

2.1 Research questions formulation

The objective of our SLR is to analyse research studies related to automated (business) process discovery. Specifically, we focused on methods that *produce process models from event logs*. This means, for example, that methods performing only trace clustering are not considered in our analysis. To this aim, we formulated the following research questions:

- RQ1 What methods exist for automated process discovery?
- RQ2 What type of process model can be discovered by these methods, and in which language?
- RQ3 Which language constructs can be captured by a model discovered by these methods?
- RQ4 What tools exist to support these methods?
- RQ5 What type of data has been used to evaluate these methods, and in which application domains?

RQ1 is the core research question, which aims at identifying existing methods to perform automated process discovery. 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., imperative, declarative or hybrid), and the specific language employed (e.g., Petri nets, BPMN, Declare). RQ3 delves into the specific language 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 which would allow us to retrieve the majority of methods in this area. Furthermore, we used "learning" and "workflow" as synonyms of "discovery", respectively, "process". This led to the following four search strings: i) "process discovery", ii) "workflow discovery", iii) "process learning", and "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. a study 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".

We applied each of the four search strings to seven popular academic databases: Scopus, Web of Science, IEEE Xplore, ACM Digital Library, SpringerLink, ScienceDirect and Google Scholar, and retrieved studies based on the occurrence of one of the search strings in the title, the keywords or the abstract of a paper. In addition, to ensure that all relevant studies were identified in our search, we performed a further search on Google Scholar by retrieving any study whose full text contained at least one of the search strings. We noted that this additional search did not return any relevant study that was not already discovered in our primary search. The search was conducted in December 2016.

Kitchenham [4] recommends to validate trial search strings against lists of already known primary studies. Accordingly, we examined an existing survey in the field of automated process discovery, namely [3], and verified that all publications contemplated in this survey were also found by our search. Given that [3] refers to studies published prior to 2012, our search returned a large number of publications not covered in [3].

2.3 Study selection

As a last step, as suggested by [5], [6], [7], [8], we defined inclusion and exclusion criteria to ensure an unbiased selection of relevant studies (to be retained, a study must satisfy all inclusion criteria and none of the exclusion criteria). The development of these criteria, as recommended in [4], was based on the objective and the scope of this survey, as defined by the research questions distilled in Section 2.1. This led to the following criteria.

Inclusion Criteria

- IN1 The study is related to automated process discovery from event logs;
- IN2 The study is published in 2011 or later;
- IN3 The study is not investigated in [3];
- IN4 The study is an improvement of a method investigated in [3].

Exclusion Criteria

- EX1 The study describes an existing method to automated process discovery without introducing any relevant improvement to it.
- EX2 The study mainly focuses on other process mining methods, e.g. conformance checking or performance mining.

EX3 The study is not peer-reviewed.
EX4 The study is not written in English.
EX5 The study describes a method for which a software implementation is not available. Conversely, if the study claims the existence of an implementation, it is provisionally included even if the implementation is not accessible.
EX6 The study describes a method that has not been evaluated.
EX7 If several studies refer to the same method, all studies except the most complete and general one are excluded.

Given that the survey in [3], published in 2012, is the most recent mapping of studies on the topic, we excluded from our search all the methods published before 2011 or already investigated in [3] (cf. IN3 and IN4).

The assessment of each study against the inclusion and exclusion criteria was performed independently by two authors of this paper. The results were compared in order to resolve inconsistencies with the mediation of a third author.

After the application of the inclusion criteria, we obtained a total of 2,165 studies. We then proceeded with the application of the exclusion criteria. First, we removed duplicates. Next, we checked if a study matched the exclusion criteria EX1-4 by analyzing title, abstract, introduction and conclusions. Further, we performed a backward reference search by considering the literature cited by the studies themselves. After this initial filtering, we obtained 330 studies.[1]

Second, we analysed in depth each of the 330 studies by focusing on EX5-6. If an approach was not implemented or evaluated, it was discarded from the SLR. This second filtering returned 98 studies. Finally, we applied EX7, so that if a method did not add any novel contribution with respect to a previous study describing the same method, or if it was a variant of one of its preceding, more general and complete method, it was excluded from the SLR too. Overall, this resulted in 32 *primary studies* describing 32 distinct automated process discovery methods.

Fig. 1 shows how the primary studies are distributed over time. We can see that the interest in the topic of automated process discovery has grown over time, with a sharp increase between years 2013 and 2014. The relatively low number of studies between 2011 and 2013 is due to the fact that most of the methods proposed in this period have later been improved upon. As discussed above, in these cases we retained the most up-to-date incarnation of each method.

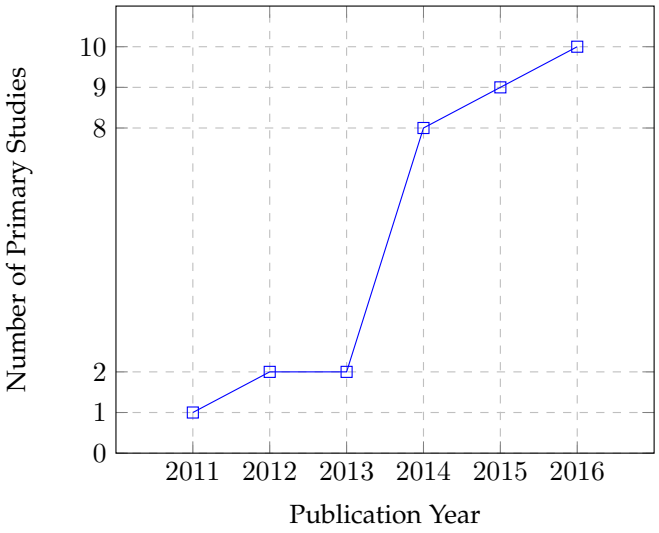


Fig. 1: Number of primary studies over time.

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) Language constructs captured in procedural models [parallelism (AND), exclusive choice (XOR), inclusive choice (OR), loop]—RQ3
3) Type of implementation (standalone or plugin, 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 domain of data (e.g. insurance, banking, healthcare)—RQ5.

This information is summarized in Table 1, where for each method we report the reference to the primary study, the year of publication and the number of citations counted at the time of the search (December 2016). Collectively, this information allows us to answer the first research question ("What methods exist for automated process discovery?"). In the remainder of this section, we proceed with surveying each method along the above classification dimensions, to answer the other research questions.

3.1 Model type and language (RQ2)

The majority of methods (25 out of 32) produce procedural models. Five approaches ([11], [12], [32], [35], [37]) discover declarative models in the form of Declare constraints, while [17] produces declarative models using the WoMan formalism. The method in [18] is the only one able to discover hybrid models as a combination of Petri nets and Declare constraints.

Moving on to the language in which the process model is represented, we can see that Petri nets is the predominant language. 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 Heuristics nets, Casual nets, State machines and simple Directed

1. The list of studies obtained in this phase is available at https://goo.gl/fq73qM

Method	Authors	Year	Citations	Model type	Model language	Procedural constructs				Implementation		Evaluation		
						AND	XOR	OR	Loop	Framework	Accessible	Real-life	Synth.	Art.
Process Spaceship	Motahari et al. [9]	2011	105	Procedural	State machines	✓	✓		✓	Eclipse	✓	✓	✓	
HK	Huang and Kumar [10]	2012	6	Procedural	Petri nets	✓	✓		✓	Standalone		✓	✓	
Declare Miner	Maggi et al. [11]	2012	32	Declarative	Declare					ProM	✓	✓	✓	
MINERful	Di Ciccio, Mecella [12]	2013	33	Declarative	Declare					ProM, Standalone	✓	✓	✓	
Inductive Miner - Infrequent	Leemans et al. [13]	2013	67	Procedural	Process trees	✓	✓	✓	✓	ProM	✓	✓		
Process Skeletonization	Abe, Kudo [14]	2014	2	Procedural	Directly-follows graphs		✓		✓	Standalone		✓		
Evolutionary Tree Miner	Buijs e al. [15]	2014	25	Procedural	Process trees	✓	✓	✓	✓	ProM	✓	✓		✓
Updated Heuristics Miner	De Cnudde et al. [16]	2014	3	Procedural	Heuristics nets	✓	✓		✓	ProM		✓		✓
WoMan	Ferilli [17]	2014	10	Declarative	WoMan					Standalone		✓		✓
Hybrid Miner	Maggi et al. [18]	2014	15	Hybrid	Declare + Petri nets	✓	✓	✓		ProM	✓	✓		
Competition Miner	Redlich et al. [19]	2014	7	Procedural	BPMN	✓	✓		✓	Standalone		✓		
Direted Acyclic Graphs	Vasilecas et al. [20]	2014	1	Procedural	Directed acyclic graphs		✓			Standalone		✓	✓	
Decomposed Process Miner	Verbeek, van der Aalst [21]	2014	7	Procedural	Petri nets	✓	✓	✓	✓	ProM	✓	✓		
MVPM Mine	Folino et al. [22]	2015	26	Procedural	Heuristics nets		✓		✓	Standalone		✓		
CNMining	Greco et al. [23]	2015	2	Procedural	Causal nets	✓	✓		✓	ProM	✓	✓	✓	
Alpha$	Guo et al. [24]	2015	2	Procedural	Petri nets	✓	✓		✓	ProM	✓	✓		✓
Maximal Pattern Mining	Liesaputra et al. [25]	2015	1	Procedural	Workflow nets	✓	✓		✓	ProM		✓		✓
Supervised Polyhedra	Ponce de Leon et al. [26]	2015	0	Procedural	Petri nets	✓	✓		✓	Standalone		✓		
DGEM	Molka et al. [27]	2015	1	Procedural	BPMN	✓	✓		✓	Standalone		✓		✓
LocalizedLogs	van der Aalst et al. [28]	2015	7	Procedural	Petri nets	✓	✓		✓	ProM	✓	✓	✓	
HybridILPMiner	van Zelst et al. [29]	2015	7	Procedural	Petri nets	✓	✓		✓	ProM	✓			✓
ProDiGen	Vazguez et al. [30]	2015	19	Procedural	Heuristics nets	✓	✓		✓	Standalone			✓	✓
Structured Miner	Augusto et al. [31]	2016	1	Procedural	BPMN	✓	✓		✓	Apromore, ProM, Stand.	✓		✓	✓
Non-Atomic Declare Miner	Bernardi et al. [32]	2016	3	Declarative	Declare					ProM	✓	✓	✓	
RegPFA	Breuker et al. [33]	2016	1	Procedural	Petri nets	✓	✓		✓	Standalone	✓	✓	✓	
BPMN Miner	Conforti et al. [34]	2016	17	Procedural	BPMN	✓	✓	✓	✓	Apromore, ProM, Stand.	✓	✓		✓
TB-MINERful	Di Ciccio et al. [35]	2016	8	Declarative	Declare					Standalone	✓	✓		✓
PGminer	Mokhov et al. [36]	2016	0	Procedural	Partial order graphs	✓	✓			Standalone, Workcraft	✓	✓		✓
SQLMiner	Schönig et al. [37]	2016	1	Declarative	Declare					Standalone	✓	✓		
ProM-D	Song et al. [38]	2016	1	Procedural	Petri nets	✓	✓		✓	Standalone		✓	✓	
CSMMiner	van Eck et al. [39]	2016	0	Procedural	State machines	✓	✓		✓	ProM	✓	✓		
Proximity Miner	Yahya et al. [40]	2016	0	Procedural	Heuristic nets	✓	✓		✓	ProM	✓	✓		

TABLE 1: Overview of the 32 primary studies resulting from the search (ordered by year and author).

Acyclic Graphs, while Declare is the most commonly-used language when producing declarative models.

Petri nets. In [10], 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.

In [21], the authors propose an improvement of the algorithm implemented in the ILP Miner [41] whose complexity is linear on the size of the event log and exponential on the number of distinct activities. The method is based on splitting up distinct activities over multiple event logs to alleviate the overall complexity. An implementation of this method is available as a ProM plugin.

The method in [24] 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 [25], the authors propose a method for automated process discovery using Maximal Pattern Mining where they construct patterns based on the whole sequence of events seen on the traces. Starting from these patterns they construct process models in the form of Workflow nets.

In [26], Ponce de Leon et al. present Supervised Polyhedra, a standalone application based on a process discovery method that takes into account negative events. This feature guarantees that the discovered models are not only simple, fitting and precise, but also good at generalizing the behavior underlying an event log.

In [28], the authors present a method for the discovery of Petri nets implemented in a ProM plug-in available in a package called LocalizedLogs. In a log, events are localized by assigning a non-empty set of regions to each event. Regions can only interact through shared events. The approach is based on a genetic algorithm that exploits localized events to improve the quality of the discovered models.

In [29], the authors propose an improvement of the traditional ILP Miner algorithm [41] 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 ILP problems during ILP-based process discovery.

The method presented in [33] allows the discovery of Petri nets using the theory of grammatical inference. The method has been implemented as a standalone application called RegPFA.

The method proposed in [38] 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.

Process trees. The Inductive Miner [13], [42], [43], [44], [45] and the Evolutionary Tree Miner [15] 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 [42]. Successively, since that method was unable to deal with infrequent behavior an upgrade was proposed in [13], which efficiently drops infrequent behavior from logs, still ensuring that the discovered model is behavioral correctness (soundness) and highly fitting. Another variant of the Inductive Miner is presented in [43]. This variant can minimize the impact of incompleteness of the input logs. Finally, the variant presented in [44] and [45] combines scalability with quality guarantees. It can be used to mine large event logs and produces sound models. The Inductive Miner is available as a ProM plugin.

In [15], Buijs et al. introduce the Evolutionary Tree Miner. This method, also implemented as a ProM plugin, 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 for a discovered model (fitness, precision, generalization and complexity).

Heuristics nets. In [46], the authors present Flexible Heuristics Miner. This method, implemented as a ProM plugin, can mine process models containing non-trivial constructs but with a low degree of block structuredness. At the same time, this method can cope well with noise in event logs. The process 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 [47]. In [16], the method presented in [46] has been improved as anomalies were found concerning the validity and completeness of the resulting process model. The improvements have been implemented in a ProM plugin called Updated Heuristics Miner.

In [22], Folino et al. propose a two-phase clustering-based process discovery method, where the clusters are inherently defined through logical decision rules over context data. The method, which also produces a Heuristics net as a result, has been implemented in a standalone tool.

ProDiGen, a standalone miner by Vazguez et al. [30], allows users to discover Heuristics 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 Heuristics nets is presented in [40]. This method extracts behavioral relations between the events of the log which are then enhanced using input from domain experts. The method has been implemented in a ProM plugin called Proximity Miner.

State machines. The authors of Process Spaceship [9] start from the observation that information about process executions is often scattered across several systems and data sources. Accordingly, they investigate different ways in which process-related events could be correlated in service interaction logs and propose a mechanism to discover event correlations (semi-)automatically from them. The data collected through event correlations is mined to discover process models in the form of state machines. This method has been implemented as an Eclipse plugin.

A second method that discovers state machines in discussed in [39]. 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 method has been implemented as ProM plugin called CSM Miner and provides an interactive visualization of these multi-perspective state-based models.

BPMN models. In [48], Conforti et al. present 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 handlings). The method has been subsequently improved in [34] to make it robust to noise in event logs. BPMN Miner is available as a standalone application and as a plugin for Apromore and ProM.

Another method to discover BPMN models is Structured Miner [31]. Different from other methods, this method separates the concern of producing accurate models with that of ensuring their block-structuredness, without sacrificing the former for the latter. Structured Miner works on top of Heuristics Miner (version 5.2) and comes as a standalone tool as well as a plugin for Apromore and ProM.

A further method to discover BPMN models is the Dynamic Constructs Competition Miner [19]. This method extends the Constructs Competition Miner presented in [49]. The method is based on a divide-and-conquer algorithm which discovers block-structured process models from logs including the discovery of exceptional behavior.

Declarative models. [11] describes a two-phase approach for mining declarative process models expressed using Declare constraints [50], [51], i.e. constraints in linear temporal logic. 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. During 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.

Another method to discover Declare constraints in MINERful [12]. This method consists of two phases as well. 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).

The WoMan framework proposed by Ferilli in [17], includes at its core a method to learn and refine process models from event logs. The method discover first-order logic constraints and guarantees incrementality in learning and adapting the models, the ability to express triggers and conditions on the process tasks and efficiency.

The method presented in [32] 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. The method has been implemented as a ProM plugin.

In [35], the authors present a method to discover a class of Declare constraints called target-branched Declare. A Declare constraint is target-branched when one of its parameters (the target) is the disjunction of two or more activities to express that one activity out of a set of activities can occur. The method has been implemented as a standalone application.

Finally, SQLMiner [37] is based on a mining approach that directly works on relational event data by querying the log with standard SQL. By leveraging database performance technology, the mining procedure is fast without limiting itself to detecting certain control-flow constraints. Queries can be customized and cover process perspectives beyond control flow [52].

Further languages. Further languages include casual nets, directly-follows graphs, directed acyclic graphs, partial order graphs and hybrid models. Greco et al. propose a

discovery method that returns casual nets [23]. A casual net is a net where only the casual relation between activities in a log is represented. The method in [23] encodes casual relations gathered from an event log and if available, background knowledge in terms of precedence constraints over the topology of the resulting process models. A discovery algorithm is formulated in terms of reasoning problems over precedence constraints.

In [14], 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, i.e. graphs that encode the direct causality relation between activities only.

Vasilecas et al. [20] 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 [36], 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 plugin and as a standalone application called PGminer.

The method in [18] puts forward the idea of discovering a hybrid model from an event log. A hybrid process model is hierarchical model, where each node represents a subprocesses which may be specified in a declarative or procedural way. They use Petri nets for representing procedural subprocesses and Declare for representing declarative subprocesses. The method has been implemented as a ProM plugin called Hybrid Miner.

3.2 Procedural language constructs (RQ3)

All the 25 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, [13], [15] are able to directly identify block-structured inclusive choices (due to using process trees), while [21], [34] can detect this construct only when used on top of the methods in [13] or [15] (i.e. indirectly).

The remaining 21 methods can discover constructs for parallelism, exclusive choice and loop, with the exception of [14], [22], which can detect exclusive choice and loop but not parallelism, and [20], which can discover exclusive choices only. This is mostly due to the nature of their outputs. Indeed, the languages supported by these methods, i.e. directly-follows graphs, heuristics nets, directed acyclic graphs and casual nets do not natively support parallelism. In other methods these languages have been extended to cater for parallelism (cf. [23], [30], [40]).

3.3 Implementation (RQ4)

Over 50% of the methods (17 out of 32) provide an implementation as a plugin for the ProM platform.[2] 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. Three of the methods which have a ProM implementation are also available as standalone tools (cf. [12], [31], [34]), while [31], [34] provide a further implementation as a plugin for Apromore,[3] an on-line process analytics platform, which is also open source, and is growing consensus among academics as a process mining tool oriented towards end users. Finally, one method [36] has been implemented as a plugin for Workcraft,[4] a platform for designing concurrent systems, and another method [9] has been implemented as a plugin for Eclipse.

Unfortunately, only a minority of tools (twelve out of 32) are made publicly available to the community. These exclude nine ProM plugins.

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 (29 out of 32) were tested using real-life logs. Among them, ten approaches (cf. [9], [10], [11], [12], [20], [23], [28], [32], [33], [38]) were further tested against synthetic logs, while nine approaches (cf. [15], [16], [17], [24], [25], [27], [34], [35], [36]) against artificial logs. Finally, two methods were tested both on synthetic and artificial logs (cf. [30], [31]), while [29] was tested on artificial 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 due to not being accessible.

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, which is held annually as part of the BPM Conference series. These logs are publicly available from the *4TU Centre for Research Data*,[5] and cover domains such as healthcare (used by [12], [13], [25], [37]), banking (used by [12], [13], [16], [18], [19], [20], [21], [27], [33], [35], [37], [39]), and IT support management in automotive (cf. [16], [22], [32], [33]) and banking (cf. [35]).

Besides these publicly-available logs, a range of private logs were also used, originating from different domains such as logistics (cf. [38], [40]), traffic congestion dynamics [23], employers habits (cf. [17], [39]), financial [28], automotive [24], administrative [26], healthcare [27], telecom [36], project management and insurance (cf. [14], [34]) and building permit approval (cf. [11], [13], [15]).

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

4 BENCHMARK

Using a selection of the methods surveyed in this paper, we conducted an extensive benchmark to identify relative advantages and trade-offs. In the rest of this section, we justify the criteria for the 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 the next section.

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 a subset. We applied the following criteria to include a method in the benchmark:

i an implementation of the method must be available and accessible
ii the implementation must take as input a simple event log (no event payload)
iii the implementation must produce as output a Petri net or a model seamlessly convertible into a Petri net.

The specific reason for enforcing these criteria is that the metrics used to evaluate the quality of the discovered model, illustrated later in this section, can only be computed on top of Petri nets.

On the basis of these criteria we selected the following methods: Inductive Miner [53], CNMining [23], $\alpha\$$ [24], Evolutionary Tree Miner [15], LocalizedLogs [28], Hybrid ILP Miner [29], Structured Miner on Heuristic Miner 5.2 [31], RegPFA [33], BPMN Miner [34], Decomposed Process Mining [21] and Heuristics Miner version 6.0 [54]. This latter method was included since it resulted to be the best among those evaluated in [3].

Methods such as Maximal Pattern Mining [25] and ProDiGen [30] were discarded because the respective implementations were not accessible, despite being presented in the papers.[6] A posteriori, we excluded Decomposed Process Mining, CNMining and BPMN Miner. We excluded Decomposed Process Mining as it follows a divide-and-conquer approach which can be applied on top of any other discovery method to improve the results. We held out CNMining because during the measurements it was not possible to convert any of its output causal nets into Petri nets, and RegPFA because its output is a Petri net only available in graphical format (DOT). Last, we excluded BPMN Miner since it produces the same results as Structured Miner in the case of flat process models (such as the ones used for this benchmark), but while the latter can only be used in conjunction with the Heuristics Miner, the former can be used on top of other discovery methods.

To conclude, the approaches selected for the benchmark are: $\alpha\$$, Inductive Miner (IM), Evolutionary Tree Miner (ETM), Heuristics Miner 6.0 (HM_6), Structured Miner over Heuristics Miner 5.2 (S-$HM_{5.2}$) and Hybrid ILP Miner (HILP).

6. We note that in these cases we contacted the authors but obtained no answer.

4.2 Setup and datasets

To guarantee the reproducibility of our benchmark and to provide the academic community with a tool for comparing new methods with the ones evaluated in this paper, we developed a command-line Java application which performs measurements of accuracy and complexity metrics on the six discovery methods selected above. The tool is open-source,[7] and can be easily extended to incorporate new discovery methods or metrics.

The dataset used for our benchmark is the collection of real-life event logs publicly available at the "4TU Centre for Research Data" as of March 2017.[8] We included all the *BPI Challenge* (BPIC) logs, plus the *Road Traffic Fines Management Process* (RTFMP) and the *SEPSIS Cases* log. They record executions of business processes in different domains, i.e. healthcare, finance, government and IT service management. We held out those logs that do not explicitly capture business processes (the BPIC 2011 and 2016 logs), and those contained in other logs (e.g. the *Environmental permit application process* log).

In three logs (BPIC14, BPIC15 and BPIC17), we applied the filtering technique in [55] to remove infrequent behavior. This was necessary since all the models discovered by the methods tested exhibited very poor accuracy (F-score close to 0 or not computable) on these logs, making the comparison useless.

Table 2 reports the characteristics of the twelve logs used. We can observe that the collection is widely heterogeneous containing both simple and very complex logs. The log size ranges from 681 traces (for the $BPIC15_{2f}$ 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%, 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, counting only one event, to very long counting 185 events.

We ran our benchmark on a 6-core Intel Xeon CPU E5-1650 v3 @ 3.50GHz with 128GB RAM. We setup Java JVM 8 with 16GB of heap space, applying a timeout of one hour for the discovery phase, and one hour for measuring each of the quality metric. Each discovery method was executed applying its default settings.

4.3 Evaluation metrics

For all selected discovery metrics 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 [56], which measures the degree to which every trace in the log can be aligned with a trace produced by the model.

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 [57], which

7. Available at https://github.com/raffaeleconforti/ResearchCode
8. https://data.4tu.nl/repository/collection:event_logs_real

Log Name	Total Traces	Distinct Traces (%)	Total Events	Distinct Events	Trace Length		
					min	avg	max
BPIC12	13087	33.4	262200	36	3	20	175
$BPIC13_{cp}$	1487	12.3	6660	7	1	4	35
$BPIC13_{inc}$	7554	20.0	65533	13	1	9	123
$BPIC14_{f}$	41353	36.1	369485	9	3	9	167
$BPIC15_{1f}$	902	32.7	21656	70	5	24	50
$BPIC15_{2f}$	681	61.7	24678	82	4	36	63
$BPIC15_{3f}$	1369	60.3	43786	62	4	32	54
$BPIC15_{4f}$	860	52.4	29403	65	5	34	54
$BPIC15_{5f}$	975	45.7	30030	74	4	31	61
$BPIC17_{f}$	21861	40.1	714198	41	11	33	113
RTFMP	150370	0.2	561470	11	2	4	20
SEPSIS	1050	80.6	15214	16	3	14	185

TABLE 2: Descriptive statistics of the event logs used in the benchmark.

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 measures the ability of an automated discovery algorithm to capture behavior that is not present in the log but that can be produced by the business process under observation. To measure generalization we use 3-fold cross validation [58]: We divide the log into three parts, discover a model from two 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 1 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, and that the discovered model is accurate and does not introduce extra behavior (i.e. does not over-generalize the behavior recorded in the log).

Complexity quantifies how difficult it is to understand a model for a model reader. Several complexity metrics have been shown to be (inversely) related to understandability [59], 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).

Lastly, *soundness* assesses the behavioral quality of a process model by reporting whether the model violates one of the three soundness criteria [60]: i) option to complete, ii) proper completion, and iii) no dead transitions.

4.4 Benchmark results

The results of the benchmark are summarized in Table 3. We highlighted in bold the best score for each measure on each log, and 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 (e.g. the model is disconnected or is unsound).

The first conclusion we can draw from these results is that four out of six discovery methods were not able to systematically discover sound models. $\alpha\$$, HM and HILP experienced severe difficulties in producing useful outputs. $\alpha\$$ showed scalability issues and it discovered only one model within the allotted discovery time of one hour, which did not stand out in accuracy (despite striking a good balance between fitness and precision) neither did it in complexity. HILP did definitely better compared to $\alpha\$$, producing a model ten times out of twelve. However, these models were either disconnected or contained multiple end places without indicating a final marking (a requirement to measure fitness and precision). For this latter reason, we could assess only the model complexity for HILP. The models are moderately simple for the $BPIC13_{cp}$, $BPIC13_{inc}$ and RTFMP logs, but highly complex for the remaining logs, with large size and CFC. HM_6 shares similar results with HILP: it always delivered an output in time, in fact faster than any other discovery method, but only two times out of twelve this output was a sound model. Precisely, for the $BPIC13_{inc}$ log, HM_6 delivered a very simple and accurate model with maximum scores for precision, F-score, size, CFC and structuredness, while for the $BPIC15_{3f}$ log, this method delivered a highly-fitting model having best fitness, F-score and generalization, thought very high complexity.

About the remaining methods, $S\text{-}HM_{5.2}$ led to slightly better results than HM_6 thanks to its ability to repair soundness issues, being able to produce five sound models out of twelve. Of these five models, $S\text{-}HM_{5.2}$ scored the best fitness, F-score and generalization on the $BPIC15_{1f}$ and $BPIC15_{5f}$ logs, being second in precision only to ETM. Also, it scored the best F-score from the RTFMP log, and delivered outputs of high quality from the two BPIC13 logs, yet not the best results.

IM and ETM were the only two methods able to systematically produce sound process models, since they discover process trees which by construction are fully block-structured and sound. Furthermore, models discovered using IM scored the highest value on fitness and generalization for nine models out of twelve, whilst models discovered by ETM scored the best results on precision for nine models as well. Also, ETM delivered five times the most accurate models (highest F-score), against two times for IM. Indeed, the lack of precision for the models produced by IM did not let it outperform ETM. It is worth mentioning that the high quality of ETM's outputs is strictly bounded to the high execution time needed by ETM's genetic algorithm (always capped at one hour). This time can be over four orders of magnitude higher than the time produced by the fastest discovery methods, which are IM and HM_6. This significant performance issue renders ETM hardly usable in practice.

Coming to the complexity of the output models, ETM and IM achieved the best results. First, both methods always output block-structured process models. Second, size and CFC are very small compared to those of other methods.

Log	Discovery Method	Accuracy			Gen.	Complexity			Sound	Exec.
		Fitness	Precision	F-score	(3-Fold)	Size	CFC	Struct.		Time(s)
BPIC12	α\$	-	-	-	-	-	-	-	-	t/o
	IM	**0.98**	0.50	**0.66**	**0.98**	**59**	37	**1.00**	yes	6.6
	ETM	0.33	**0.98**	0.49	0.38	69	**10**	**1.00**	yes	3,600
	HM_6	-	-	-	-	85	99	0.05	no	**2.5**
	$S\text{-}HM_{5.2}$	-	-	-	-	128	177	0.10	no	341.0
	HILP	-	-	-	-	300	460	-	no	772.2
$BPIC13_{cp}$	α\$	-	-	-	-	-	-	-	-	t/o
	IM	0.82	**1.00**	**0.90**	0.82	**9**	4	**1.00**	yes	**0.1**
	ETM	**0.99**	0.76	0.86	**0.99**	11	17	**1.00**	yes	3,600
	HM_6	-	-	-	-	12	6	0.67	no	**0.1**
	$S\text{-}HM_{5.2}$	0.83	0.79	0.81	0.86	17	13	**1.00**	yes	1.5
	HILP	-	-	-	-	10	**3**	-	yes	**0.1**
$BPIC13_{inc}$	α\$	-	-	-	-	-	-	-	-	t/o
	IM	**0.92**	0.54	0.68	**0.92**	13	7	**1.00**	yes	1.0
	ETM	0.84	0.80	0.82	0.88	28	24	**1.00**	yes	3,600
	HM_6	0.91	**0.96**	**0.93**	0.91	**9**	**4**	**1.00**	yes	**0.8**
	$S\text{-}HM_{5.2}$	0.83	0.95	0.89	0.83	16	11	**1.00**	yes	12.8
	HILP	-	-	-	-	24	9	-	yes	2.5
$BPIC14_f$	α\$	-	-	-	-	-	-	-	-	t/o
	IM	**0.89**	0.64	0.74	**0.89**	31	18	**1.00**	yes	3.4
	ETM	0.68	**0.94**	**0.79**	0.57	**22**	**15**	**1.00**	yes	3,600
	HM_6	-	-	-	-	43	51	-	no	**3.3**
	$S\text{-}HM_{5.2}$	-	-	-	-	35	24	-	no	47.9
	HILP	-	-	-	-	80	59	-	no	7.3
$BPIC15_{1f}$	α\$	0.71	0.73	0.76	t/o	219	91	0.22	yes	3,545.9
	IM	0.97	0.57	0.71	0.96	164	108	**1.00**	yes	0.6
	ETM	0.57	**0.89**	0.69	0.56	**73**	**21**	**1.00**	yes	3,600
	HM_6	-	-	-	-	150	98	-	no	**0.5**
	$S\text{-}HM_{5.2}$	**1.00**	0.75	**0.86**	**1.00**	145	89	0.39	yes	481.5
	HILP	-	-	-	-	282	322	-	no	4.4
$BPIC15_{2f}$	α\$	-	-	-	-	-	-	-	-	t/o
	IM	**0.93**	0.56	0.70	**0.94**	193	123	**1.00**	yes	**0.7**
	ETM	0.62	**0.90**	0.73	0.57	**78**	**19**	**1.00**	yes	3,600
	HM_6	-	-	-	-	194	158	0.11	no	**0.7**
	$S\text{-}HM_{5.2}$	-	-	-	-	196	154	0.11	no	119.8
	HILP	-	-	-	-	-	-	-	-	t/o
$BPIC15_{3f}$	α\$	-	-	-	-	-	-	-	-	t/o
	IM	**0.95**	0.55	0.70	**0.95**	159	108	**1.00**	yes	1.3
	ETM	0.66	**0.88**	0.75	0.64	**78**	**26**	**1.00**	yes	3,600
	HM_6	**0.95**	0.67	**0.79**	**0.95**	157	151	0.07	yes	**0.8**
	$S\text{-}HM_{5.2}$	-	-	-	-	184	183	0.06	no	1,246.6
	HILP	-	-	-	-	433	829	-	no	1,062.9
$BPIC15_{4f}$	α\$	-	-	-	-	-	-	-	-	t/o
	IM	**0.96**	0.58	0.73	**0.96**	162	111	**1.00**	yes	0.7
	ETM	0.66	**0.95**	**0.78**	0.63	**74**	**17**	**1.00**	yes	3,600
	HM_6	-	-	-	-	156	127	0.13	no	**0.5**
	$S\text{-}HM_{5.2}$	-	-	-	-	155	123	0.16	no	301.7
	HILP	-	-	-	-	364	593	-	no	14.7
$BPIC15_{5f}$	α\$	-	-	-	-	-	-	-	-	t/o
	IM	0.94	0.18	0.30	0.94	134	95	**1.00**	yes	1.5
	ETM	0.58	**0.89**	0.70	0.56	**82**	**26**	**1.00**	yes	3,600
	HM_6	-	-	-	-	166	124	0.15	no	**1.2**
	$S\text{-}HM_{5.2}$	**1.00**	0.70	**0.82**	**1.00**	167	123	0.20	yes	304.0
	HILP	-	-	-	-	-	-	-	-	t/o
$BPIC17_f$	α\$	-	-	-	-	-	-	-	-	t/o
	IM	**0.98**	0.70	0.82	**0.98**	35	20	**1.00**	yes	13.3
	ETM	0.72	**1.00**	**0.84**	0.82	31	**5**	**1.00**	yes	3,600
	HM_6	-	-	-	-	**29**	10	0.45	no	**6.5**
	$S\text{-}HM_{5.2}$	-	-	-	-	195	114	0.68	no	390.9
	HILP	-	-	-	-	222	330	-	no	384.5
RTFMP	α\$	-	-	-	-	-	-	-	-	t/o
	IM	**0.99**	0.70	0.82	**0.99**	**34**	**20**	**1.00**	yes	10.9
	ETM	0.79	**0.98**	0.87	0.81	46	33	**1.00**	yes	3,600
	HM_6	-	-	-	-	47	50	0.06	no	7.8
	$S\text{-}HM_{5.2}$	0.92	0.88	**0.90**	0.77	70	55	**1.00**	yes	260.9
	HILP	-	-	-	-	57	53	-	no	**3.5**
SEPSIS	α\$	-	-	-	-	-	-	-	-	t/o
	IM	**0.99**	0.45	0.62	**0.96**	50	32	**1.00**	yes	0.4
	ETM	0.71	**0.84**	**0.77**	0.70	**30**	**15**	**1.00**	yes	3,600
	HM_6	-	-	-	-	81	132	0.17	no	**0.03**
	$S\text{-}HM_{5.2}$	-	-	-	-	52	42	0.58	no	312.3
	HILP	-	-	-	-	87	129	-	no	1.6

TABLE 3: Results of the benchmark.

However, ETM tends to output more compact models than IM, having the smallest size and CFC seven times out of twelve.

To conclude, while there is no clear winner, IM provided the best results altogether, followed by ETM, S-$HM_{5.2}$ and HM_6, with the latter two methods being able to produce good quality models only from simpler logs out. And while ETM exhibited similar results to IM in terms of quality, it required significantly longer execution times.

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 addressing the problem of this area of literature. Despite the variety of proposals we can clearly identify two main streams: methods which output procedural process models and methods which output declarative process models. Further, while the latter only relies on declarative statements to represent a process, the former provides various language alternatives, though, most of these methods output Petri nets. The predominance of Petri nets is driven by the expressiveness 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 in any case, have a higher practical relevance than their declarative counterparts, given that declarative process models are hardly used in practice.

Finally, our benchmark shows gaps, limitations and unexplored trade-offs of procedural methods for automated process discovery. These include lack of scalability to large and complex logs, and strong differences in the output models, across the various quality metrics. On this latter aspect, the majority of methods were not able to guarantee soundness, except for IM and ETM, which do so at the price of enforcing block-structuredness. Moreover, ETM requires significantly longer execution times than IM, rendering this method hardly usable in practice. Further, our evaluation shows that existing methods do not integrate sufficiently robust noise-filtering techniques to be used on large-scale real-life events logs. In some cases (i.e. BPIC14, BPIC15, BPIC17 logs), in order to produce any output for which fitness and precision could be measured or to avoid extreme over-fitting (in the case of IM), we had to apply a noise-filtering technique as a pre-processing step.

To conclude, even if many proposals are available in this research area, there is no definitive solution to the problem of automated process discovery. First, the great majority of the methods do not have a working or available implementation, which hampers their systematic evaluation, so one can only rely on the results reported in the respective papers. Second, for those methods we assessed, we were not able to identify a clear winner, since all the evaluated methods struggle to strike a good balance between fitness, precision and complexity. They either maximize fitness at the expenses of precision, or achieve a good level of fitness and precision at the price of obtaining a very complex, often unsound model. Despite these considerations, it can be noted that there has been significant progress in this field in the past five years. Three of the methods proposed since 2011 (i.e. IM, ETM and S-$HM_{5.2}$) clearly outperform those that were developed in the previous decade.

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 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 that can be directly translated to Petri nets). Also, it only considers major studies identified in the SLR, with at least ten citations and an available implementation, and takes the most recent incarnation of each method (e.g. we only considered α\$, as it was shown in previous work to outperform the α and $\alpha+$ algorithms). 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 twelve 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 characteristics, including different application domains. 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.

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 altogether, all of which are included in the 330 studies we identified during our systematic literature review (prior to filtering).

The benchmark reported in [3] includes seven approaches, namely AGNEsMiner, $\alpha+$, $\alpha++$, Genetic Miner (and a variant thereof), Flower Heuristics Miner and ILP Miner. In comparison, our benchmark includes α\$ (which is an improved version of $\alpha+$ and $\alpha++$), Heuristics miner and HILP (the latter being an improved version of ILP). We did

not include AGNEsMiner and Genetic Miner due to the very long execution times (in the order of several hours to several days as reported in [3]). We did include however ETM in our benchmark, which is a genetic algorithm postdating the evaluation in [3], which achieves better execution times by focusing on block-structured process models and simpler transformation rules.

Another difference with respect to [3] is that in our paper we restricted the evaluation to public event logs in order to ensure reproducibility, whereas the evaluation in [3] is solely based on closed datasets due to the unavailability of public datasets at the time that study.

In terms of the findings, [3] found that Heuristics 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 other techniques postdating these ones, specifically IM and ETM, achieve even better F-score and lower model complexity than Heuristics Miner, which in addition suffers from the fact that it often produces unsound process models. It thus appears that progress in the field in the last years has been achieved by focusing on the discovery of block-structured process models.

Another previous survey in the field is outdated [61] and a more recent one is not intended to be comprehensive [62], but rather focuses on plugins 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 [63]. The methods for the evaluation of fitness and precision used in our benchmark are taken from this suite.

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 and eight quality metrics. The toolset used in this benchmark is available as open-source software and all the event logs are sourced from the 4TU Centre. 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 (330 relevant papers were identified). Traditionally, many of these proposals produce Petri nets, but more recently, we observe an increasing number of methods that produce models in other languages, including BPMN and declarative constraints. We also observe a recent emphasis on methods that produce block-structured process models.

The results of the empirical evaluation show that methods that seek to produce block-structured process models (IM, ETM and S-HM$_{5.2}$) achieve the best performance in terms of fitness, precision and complexity. Methods that do not restrict the topology of the generated process models (α\$, Heuristics Miner and HILP), generally produce unsound process models when applied to real-life event logs. We have 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. None of the methods appears to have a sufficiently powerful and adaptive filtering technique to cope with the complexity of these event logs.

Finally, the study has shown that there is still significant room for improvement in the field. One of the two top-performing automated process discovery methods (IM) is problematic because it often generates "flower" structures, i.e. fragments where a set of activities can be performed any number of times and in any order. These structures lead to low precision. The other top-performing method (ETM) suffers from significant performance issues, relying on a genetic algorithm at its core. Finally, S-HM$_{5.2}$, a relatively well-performing method, is not robust – it sometimes produces unsound process models due to its reliance on the Heuristics Miner.

Acknowledgments

This research is partly funded by the Australian Research Council (grant DP150103356) and the Estonian Research Council (grant IUT20-55).

References

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

[2] W. van der Aalst, *Process Mining: Data Science in Action*. 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," *Information Systems*, vol. 37, no. 7, pp. 654–676, 2012.

[4] B. Kitchenham, "Procedures for performing systematic reviews," *Keele, UK, Keele University*, vol. 33, no. 2004, pp. 1–26, 2004.

[5] A. Fink, *Conducting research literature reviews: from the internet to paper*, 3rd ed. Sage Publications, 2010.

[6] C. Okoli and K. Schabram, "A guide to conducting a systematic literature review of information systems research," *Sprouts: Working Papers on Information Systems*, vol. 10, no. 26, pp. 1–49, 2010.

[7] J. Randolph, "A guide to writing the dissertation literature review," *Practical Assessment, Research & Evaluation*, vol. 14, no. 13, pp. 1–13, 2009.

[8] R. Torraco, "Writing integrative literature reviews: guidelines and examples," *Human Resource Development Review*, vol. 4, no. 3, pp. 356–367, 2005.

[9] H. R. Motahari-Nezhad, R. Saint-Paul, F. Casati, and B. Benatallah, "Event correlation for process discovery from web service interaction logs," *The VLDB Journal—The International Journal on Very Large Data Bases*, vol. 20, no. 3, pp. 417–444, 2011.

[10] Z. Huang and A. Kumar, "A study of quality and accuracy trade-offs in process mining," *INFORMS Journal on Computing*, vol. 24, no. 2, pp. 311–327, 2012.

[11] 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 *Advanced Information Systems Engineering - 24th International Conference, CAiSE 2012, Gdansk, Poland, June 25-29, 2012. Proceedings*, 2012, pp. 270–285.

[12] C. Di Ciccio and M. Mecella, "A two-step fast algorithm for the automated discovery of declarative workflows," in *Computational Intelligence and Data Mining (CIDM), 2013 IEEE Symposium on*. IEEE, 2013, pp. 135–142.

[13] 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 *Business Process Management Workshops - BPM 2013 International Workshops, Beijing, China, August 26, 2013, Revised Papers*, 2013, pp. 66–78.

[14] M. Abe and M. Kudo, "Business monitoring framework for process discovery with real-life logs," in *International Conference on Business Process Management*. Springer, 2014, pp. 416–423.

[15] J. C. Buijs, B. F. van Dongen, and W. M. van der Aalst, "Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity," *International Journal of Cooperative Information Systems*, vol. 23, no. 01, p. 1440001, 2014.

[16] S. De Cnudde, J. Claes, and G. Poels, "Improving the quality of the heuristics miner in prom 6.2," *Expert Systems with Applications*, vol. 41, no. 17, pp. 7678–7690, 2014.

[17] S. Ferilli, "Woman: logic-based workflow learning and management," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 6, pp. 744–756, 2014.

[18] F. M. Maggi, T. Slaats, and H. A. Reijers, "The automated discovery of hybrid processes," in *International Conference on Business Process Management*. Springer, 2014, pp. 392–399.

[19] D. Redlich, T. Molka, W. Gilani, G. Blair, and A. Rashid, "Dynamic constructs competition miner-occurrence-vs. time-based ageing," in *International Symposium on Data-Driven Process Discovery and Analysis*. Springer, 2014, pp. 79–106.

[20] O. Vasilecas, T. Savickas, and E. Lebedys, "Directed acyclic graph extraction from event logs," in *International Conference on Information and Software Technologies*. Springer, 2014, pp. 172–181.

[21] H. Verbeek and W. M. van der Aalst, "Decomposed process mining: The ilp case," in *International Conference on Business Process Management*. Springer, 2014, pp. 264–276.

[22] F. Folino, M. Guarascio, and L. Pontieri, "On the discovery of explainable and accurate behavioral models for complex lowly-structured business processes." in *ICEIS (1)*, 2015, pp. 206–217.

[23] G. Greco, A. Guzzo, F. Lupia, and L. Pontieri, "Process discovery under precedence constraints," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 9, no. 4, p. 32, 2015.

[24] Q. Guo, L. Wen, J. Wang, Z. Yan, and S. Y. Philip, "Mining invisible tasks in non-free-choice constructs," in *International Conference on Business Process Management*. Springer, 2015, pp. 109–125.

[25] V. Liesaputra, S. Yongchareon, and S. Chaisiri, "Efficient process model discovery using maximal pattern mining," in *International Conference on Business Process Management*. Springer, 2015, pp. 441–456.

[26] H. Ponce-de Léon, J. Carmona, and S. K. Vanden Broucke, "Incorporating negative information in process discovery," in *International Conference on Business Process Management*. Springer, 2015, pp. 126–143.

[27] T. Molka, D. Redlich, M. Drobek, X.-J. Zeng, and W. Gilani, "Diversity guided evolutionary mining of hierarchical process models," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM, 2015, pp. 1247–1254.

[28] W. M. van der Aalst, A. Kalenkova, V. Rubin, and E. Verbeek, "Process discovery using localized events," in *International Conference on Applications and Theory of Petri Nets and Concurrency*. Springer, 2015, pp. 287–308.

[29] S. J. van Zelst, B. F. van Dongen, and W. M. P. van der Aalst, "ILP-Based Process Discovery Using Hybrid Regions," in *International Workshop on Algorithms & Theories for the Analysis of Event Data, ATAED 2015*, ser. CEUR Workshop Proceedings, vol. 1371. CEUR-WS.org, 2015, pp. 47–61.

[30] B. Vázquez-Barreiros, M. Mucientes, and M. Lama, "Prodigen: Mining complete, precise and minimal structure process models with a genetic algorithm," *Information Sciences*, vol. 294, pp. 315–333, 2015.

[31] A. Augusto, R. Conforti, M. Dumas, M. La Rosa, and G. Bruno, "Automated discovery of structured process models: Discover structured vs. discover and structure," in *Conceptual Modeling: 35th International Conference, ER 2016, Gifu, Japan, November 14-17, 2016, Proceedings 35*. Springer, 2016, pp. 313–329.

[32] 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?" *Information Systems*, 2016.

[33] D. Breuker, M. Matzner, P. Delfmann, and J. Becker, "Comprehensible predictive models for business processes," *MIS Quarterly*, vol. 40, no. 4, pp. 1009–1034, 2016.

[34] R. Conforti, M. Dumas, L. García-Bañuelos, and M. La Rosa, "Bpmn miner: Automated discovery of bpmn process models with hierarchical structure," *Information Systems*, vol. 56, pp. 284–303, 2016.

[35] C. Di Ciccio, F. M. Maggi, and J. Mendling, "Efficient discovery of target-branched declare constraints," *Information Systems*, vol. 56, pp. 258–283, 2016.

[36] 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*. Springer, 2016, pp. 114–136.

[37] S. Schönig, A. Rogge-Solti, C. Cabanillas, S. Jablonski, and J. Mendling, "Efficient and customisable declarative process mining with sql," in *International Conference on Advanced Information Systems Engineering*. Springer, 2016, pp. 290–305.

[38] W. Song, H.-A. Jacobsen, C. Ye, and X. Ma, "Process discovery from dependence-complete event logs," *IEEE Transactions on Services Computing*, vol. 9, no. 5, pp. 714–727, 2016.

[39] M. L. van Eck, N. Sidorova, and W. M. van der Aalst, "Discovering and exploring state-based models for multi-perspective processes," in *International Conference on Business Process Management*. Springer, 2016, pp. 142–157.

[40] B. N. Yahya, M. Song, H. Bae, S.-o. Sul, and J.-Z. Wu, "Domain-driven actionable process model discovery," *Computers & Industrial Engineering*, 2016.

[41] J. M. E. M. van der Werf, B. F. van Dongen, C. A. J. Hurkens, and A. Serebrenik, "Process discovery using integer linear programming," *Fundam. Inform.*, vol. 94, no. 3-4, pp. 387–412, 2009.

[42] 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 *Application and Theory of Petri Nets and Concurrency - 34th International Conference, PETRI NETS 2013, Milan, Italy, June 24-28, 2013. Proceedings*, 2013, pp. 311–329.

[43] S. J. Leemans, D. Fahland, and W. M. van der Aalst, "Discovering block-structured process models from incomplete event logs," in *International Conference on Applications and Theory of Petri Nets and Concurrency*. Springer, 2014, pp. 91–110.

[44] ——, "Scalable process discovery with guarantees," in *International Conference on Enterprise, Business-Process and Information Systems Modeling*. Springer, 2015, pp. 85–101.

[45] ——, "Scalable process discovery and conformance checking," *Softw. Syst. Model.*, 2017, to appear.

[46] A. J. M. M. Weijters and J. T. S. Ribeiro, "Flexible heuristics miner (FHM)," in *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2011, part of the IEEE Symposium Series on Computational Intelligence 2011, April 11-15, 2011, Paris, France*, 2011, pp. 310–317.

[47] A. J. M. M. Weijters and W. M. P. van der Aalst, "Rediscovering workflow models from event-based data using little thumb," *Integrated Computer-Aided Engineering*, vol. 10, no. 2, pp. 151–162, 2003.

[48] R. Conforti, M. Dumas, L. García-Bañuelos, and M. La Rosa, "Beyond tasks and gateways: discovering bpmn models with sub-processes, boundary events and activity markers," in *International Conference on Business Process Management*. Springer, 2014, pp. 101–117.

[49] D. Redlich, T. Molka, W. Gilani, G. S. Blair, and A. Rashid, "Constructs competition miner: Process control-flow discovery of bp-domain constructs," in *Business Process Management - 12th International Conference, BPM 2014, Haifa, Israel, September 7-11, 2014. Proceedings*, 2014, pp. 134–150.

[50] M. Pesic, H. Schonenberg, and W. M. P. van der Aalst, "DECLARE: full support for loosely-structured processes," in *11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007), 15-19 October 2007, Annapolis, Maryland, USA*, 2007, pp. 287–300.

[51] M. Westergaard and F. M. Maggi, "Declare: A tool suite for declarative workflow modeling and enactment," in *Proceedings of the Demo Track of the Nineth Conference on Business Process Management 2011, Clermont-Ferrand, France, August 31st, 2011*, 2011.

[52] S. Schönig, C. Di Ciccio, F. M. Maggi, and J. Mendling, "Discovery of multi-perspective declarative process models," in *Service-Oriented Computing - 14th International Conference, ICSOC 2016, Banff, AB, Canada, October 10-13, 2016, Proceedings*, 2016, pp. 87–103.

[53] S. J. Leemans, D. Fahland, and W. M. van der Aalst, "Discovering block-structured process models from event logs containing infrequent behaviour," in *International Conference on Business Process Management*. Springer, 2013, pp. 66–78.

[54] A. Weijters and J. Ribeiro, "Flexible heuristics miner (fhm)," in *Computational Intelligence and Data Mining (CIDM), 2011 IEEE Symposium on*. IEEE, 2011, pp. 310–317.

[55] R. Conforti, M. L. Rosa, and A. ter Hofstede, "Filtering out infrequent behavior from business process event logs," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 2, 2017.

[56] A. Adriansyah, B. van Dongen, and W. van der Aalst, "Conformance checking using cost-based fitness analysis," in *Proc. of EDOC*. IEEE, 2011.
[57] A. Adriansyah, J. Munoz-Gama, J. Carmona, B. F. van Dongen, and W. M. P. van der Aalst, "Alignment based precision checking," in *Proc. of BPM Workshops*, ser. LNBIP, vol. 132. Springer, 2012, pp. 137–149.
[58] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *International Joint Conference on Artificial Intelligence, IJCAI*. Morgan Kaufmann, 1995, pp. 1137–1145.
[59] J. Mendling, *Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness*. Springer, 2008.
[60] W. M. P. van der Aalst, *Verification of workflow nets*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 407–426.
[61] van der Aalst, W. M. P., van Dongen, B. F., J. Herbst, L. Maruster, G. Schimm, and Weijters, A. J. M. M., "Workflow mining: a survey of issues and approaches," *Data Knowl. Eng.*, vol. 47, no. 2, pp. 237–267, 2003.
[62] J. Claes and G. Poels, "Process Mining and the ProM Framework: An Exploratory Survey," in *Business Process Management Workshops*. Springer, 2012, pp. 187–198.
[63] S. K. L. M. vanden Broucke, J. D. Weerdt, J. Vanthienen, and B. Baesens, "A comprehensive benchmarking framework (CoBeFra) for conformance analysis between procedural process models and event logs in ProM," in *IEEE Symposium on Computational Intelligence and Data Mining, CIDM*. IEEE, 2013, pp. 254–261.

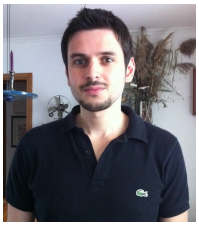
Adriano Augusto is a joint-PhD student at University of Tartu (Estonia) and Queensland University of Technology (Australia). He graduated in Computer Engineering at Polytechnic of Turin (Italy) in 2016, presenting a master thesis in the field of Process Mining.

Raffaele Conforti is Post-Doctoral Research Fellow at the Queensland University of Technology, 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 Professor of Software Engineering at University of Tartu, Estonia. Prior to this appointment he was faculty member at Queensland University of Technology and visiting researcher at SAP Research, Australia. 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, 2013).

Marcello La Rosa is Professor of Information Systems at the Queensland University of Technology, Australia. His research interests include process mining, consolidation and automation. He leads the Apromore Initiative (www.apromore.org), a strategic collaboration between various universities for the development of a process analytics platform, and co-authored the textbook "Fundamentals of Business Process Management" (Springer, 2013).

Fabrizio Maria Maggi is a Senior Researcher at the University of Tartu, Estonia. He worked as Post-Doctoral Researcher at the Department of Mathematics and Computer Science, Eindhoven University of Technology. His research interest span business process management, data mining and service-oriented computing.



Andrea Marrella is Post-Doctoral Research Fellow at the Department of Computer, Control, and Management Engineering at Sapienza University of Rome. His research interests include Human-Computer Interaction, User Experience Design, Knowledge Representation, Reasoning about Action, Automated Planning, Business Process Management. He has published over 40 research papers and articles and 1 book chapter on the above topics, among others in ACM Transactions on Intelligent Systems and Technologies, IEEE Internet Computing and Journal on Data Semantics.

Massimo Mecella is Associate Professor (with a qualification to Full Professorship) with Sapienza Università di Roma, Department of Engineering in Computer, Control and Management Sciences. His research focuses on service oriented computing, business process management, Cyber-Physical Systems and Internet-of-Things, advanced interfaces and human-computer interaction, with applications in fields such as eGovernment, eBusiness, smart houses and smart spaces and healthcare. He published more than 150 research papers and chaired different conferences in the above areas.

Allar Soo Allar Soo is student in the Masters of Software Engineering at University of Tartu. His Masters thesis is focused on automated process discovery and its use in practical settings.