Metrics in process discovery

Fabian Rojas Blum

Computer Science Department, Universidad de Chile, Chile.

Abstract

The field of process mining aims at extracting knowledge from event logs. It has caught the attention of many developers, scientists, researchers and organizations that conform an active and growing community. This has lead to an increasing number of techniques being developed. However, it is hard to evaluate the results of these techniques because there is a recognized lack of frameworks and benchmarks for comparison and evaluation. Process discovery is a subfield of process mining whose focus is on the extraction of real process models (as-is). This survey gives a background on the field and groups the metrics that have been proposed to measure the results of different techniques w.r.t process discovery techniques. Moreover, it presents the range of values of these metrics reported in the literature and their implication in the resulting models extracted with process discovery techniques.

1 Introduction

The idea of process mining (PM) is to discover, monitor and improve real or actual processes (as-is), instead of an assumed or desired processes (to-be). For this purpose, logs are the cornerstone needed for extracting valuable knowledge about processes [19]. Those logs come from a variety of sources (e.g., BPM/Workflow systems, ERP, CRM, Messaging). Moreover, with the rapid growth of the Internet of Events (IoE) and therefore the Internet of Things (IoT), more and more data is readily available [20]. This data can be processed using PM techniques to extract valuable information. However, it is not generally in a standardized format (e.g., XES [26] or MXML [24]) and therefore it has to be extracted, refined, filtered and converted into event logs. PM techniques can be used to automatically discover processes from logs; this is the subarea of process discovery and it is the main focus of this survey. PM can also be used to find and understand the reasons for bottlenecks, detect and understand deviations, predict costs, risks and delays, recommend actions, support redesign or simulation on process models [19].

Even though some of the ideas of PM are not new [1] [5], they were improved, extended and gathered to conform what now is known as PM. It can be seen as

the "missing link" between data mining and traditional model-driven Business Process Management (BPM) [21]. Furthermore, PM has reached a maturity level through a growing community of researchers and the acceptance of the proposed techniques in a variety of application domains. Moreover, the IEEE has established a Task Force on Process Mining, which promotes the research, development, education and comprehension on process mining. This task force drafted the process mining manifesto in 2011 [21], which gathered the state of the art in PM, and identified challenges and guidelines for the application of PM techniques. This manifesto also defines four criteria for judging the quality of PM results: fitness, simplicity, precision and generalization. These criteria is defined in Section 3.

Process models are a valuable asset for an organization since they are used for a variety of purposes, which will be discussed in Section 2.1. Moreover, unlike most BPM approaches, the process models mined by process discovery are driven by factual event data (logs) rather than hand-made models. Thus, the resulting process models reflect what is really done by the organization (as-is), instead of idealized (to-be) process models. Therefore, the mined process models may be very valuable assets for an organization [19]. However, for a model to be useful it is important that mined models reflect the behavior seen in the log in an appropriate way, neither overfitting nor underfitting the model and they must be as simple as possible to be more understandable. To ensure this, metrics were defined to assess the quality of the current variety of process discovery algorithms [23].

This survey was made following the concepts and guides from [9] and [31]. In this regard, two research questions were made: RQ1: What metrics have been used to measure the quality of process discovery results?, and RQ2: What are the ranges found in the literature for these metrics? Based on these questions, some keywords were selected: "Process mining", "Process discovery", "metric*", "measure*", "evaluat*", "framework", "benchmark". After that, 4 databases were consulted: ACM Digital Library, IEEE Xplore, Springer, Science Direct with papers containing the first three keywords; the other keywords were found to be too general and were only used to search by title. Then, an exclusion criterion was used in two phases. First, papers that were duplicated, in a language other than English or not fully available were excluded. Second, the rest of the papers were evaluated and classified according to their title and abstract in three groups: "yes", "no", "partially". These groups correspond to relevant, not relevant and partially relevant respectively. Finally, this survey is based on papers identified as relevant and partially relevant, as well as some papers resulting from snowballing.

The main contributions of this survey are the answers to the research questions. RQ1 aims at gathering, discussing and reporting the metrics that have been used to measure the results of PM techniques in the subfield of process discovery. These metrics can be used to measure new techniques being developed or the ones that were not appropriately evaluated. Regarding RQ2, let us say that we have built a new process discovery technique and measured it with any metric obtaining a value. How can I compare it with other discovery

techniques? This survey reports the range of values found for these metrics that can serve as a reference.

The survey is divided as follows: Section 2 gives an overview of process discovery and related concepts that are necessary to understand the rest of this work. Section 3 shows the 4 quality criteria defined in the manifesto which are used to classify the metrics. Section 4 reports the set of process discovery evaluation metrics that were reported in the literature. Section 5 gives some conclusions and discusses the two research questions addressed by this survey. Finally, this survey ends with two appendices: Appendix A shows the formulas or steps required to calculate the metrics discussed in Section 4. Appendix B contains a table that summarizes the metrics classified by quality criteria and for each one, the interval of values that was found in the literature.

2 Background

This section gives an overview of the concepts needed to discuss PM. First, the applications of Process Modeling are introduced along with the most used notation in process mining which is Petri nets. Second, event logs are discussed since they are fundamental for any PM technique. Finally, this section briefly explains the subfield of process discovery which is the focus on this survey.

2.1 Process modeling

Even though some organizations may use only informal processes, or may not even define processes at all, it is essential for organizations with a higher level of maturity to have defined and documented process models. Process models are mainly used for [19]:

- Documentation: processes are documented for instructing people or certification purposes (e.g., ISO 9000 quality management, CMMI).
- Discussion: the stakeholders use models to structure discussions.
- Verification: models are analyzed to find errors in systems or procedures (e.g., potential deadlocks)
- Performance analysis: techniques like simulation can be used to understand the factors influencing response times, service levels, etc.
- Overview: view the process from various points of view or levels of abstraction for different stakeholders.

There is a plethora of process modeling notations (e.g., Petri nets, causal nets, BPMN, activity diagrams, transition systems, YAWL, EPCs, etc.) In this work, we focus on Petri nets, as well as Workflow nets, a special subclass of Petri nets. Workflow nets are the most widely used notation in the process discovery field.

2.1.1 Petri Nets

In this work, we focus on Petri nets, as well as a subclass of them called Workflow-nets. They are a mathematical modeling languages for processes [14]. Due to its solid background, properties and intuitive graphical notation, that is able to express choice, iteration, and concurrent execution, it has been widely used for process modeling and specifically PM. Petri nets are directed bipartite graphs with two types of nodes (places and transitions), arcs and tokens. Figure 1 shows three examples of Petri nets. Generally, transitions (rectangles) represent activities, places (circles) and arcs (edges that connect transitions with places and viceversa) represent the process flow with its corresponding constructs (e.g., AND, XOR joins and splits). Finally tokens, which define the marking of the Petri net, are represented as filled circles inside places, they move over the model indicating the activities that are enabled and thus can be executed through the so called *firing rule*: A transition is *enabled* if each of its input places has a token; an enabled transition can be executed or *fired*, consuming one token from each input place (also represented as $\{\bullet p\}$), and producing one token for each output place (also represented as $\{p\bullet\}$). A *firing sequence* is a set of transitions that correspond to the firing of consecutive enabled transitions from an initial marking.

Even though it is common to use letters to identify transitions, some of them correspond to an activity. Activities are encoded with letters for conciseness. A so called *Labeled Petri net* also contains a set of activity names and a function ℓ that maps a transition to an activity name. As shown in [30], some processes also need the inclusion of the so called *invisible or silent tasks* in the model for routing purposes. In this case, labeled Petri nets reserve the label τ, i.e., $l(t) = \tau$. Thus, a Marked Petri net can be seen as a pair (N, M) [19], where:

N is a tuple (P, T, F) which represents the Petri net.
P is a finite set of places.
T is a finite set of transitions such that $P \cap T = \emptyset$.
F is a set of directed arcs such that $F \subseteq (P \times T) \cup (T \times P)$.
M is a multiset over P which denotes its current marking.

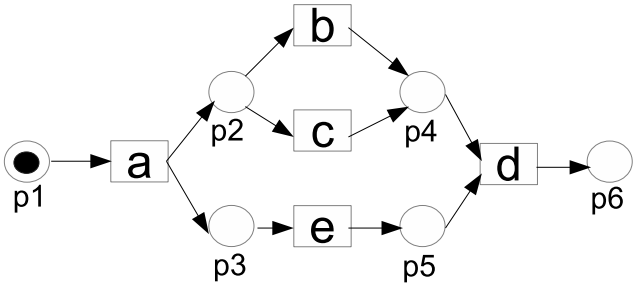
For example, the Petri net in Figure 1a shows a Marked Petri net where:

$P = \{p_1, p_2, p_3, p_4, p_5, p_6\}$.
$T = \{a, b, c, d, e\}$.
$F = \{(p_1, a), (a, p_2), (a, p_3), (p_2, b), (p_2, c), (p_3, e), (b, p_4),$
$(c, p_4), (e, p_5), (p_4, d), (p_5, d), (d, p_6)\}$.
$M = [p_1]$.

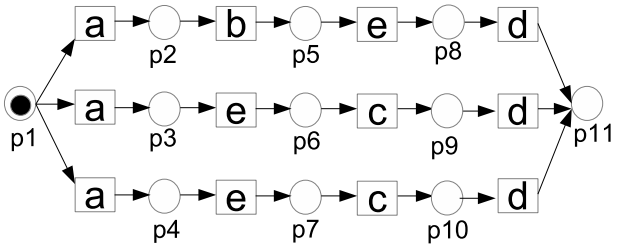
The unique enabled transition is a. Firing a consumes the token in p_1 and produces tokens in p_2 and p_3. An example of a valid firing sequence is: a, b, e, d.

2.1.2 Workflow Nets

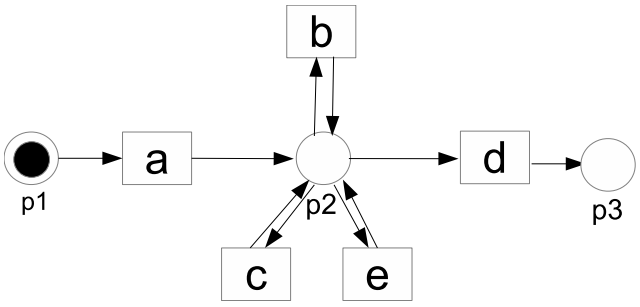
A Workflow net [18] (WF-net) is a Petri net with 3 additional requirements:



(a) M_1 Normal model



(b) M_2 Enumerative model



(c) M_3 Flower model

Figure 1: Different process models as Petri nets

- It has a source place i (also known as initial place) where the process starts such that it has no input places (i.e., $\bullet i = \emptyset$).
- It has a sink place o (also known as final place) where the process ends such that it has no output places (i.e., $o\bullet = \emptyset$).
- The addition of a silent transition with input place i and output place o (also known as the short-circuited net) results in a strongly connected graph.(i.e., there is a directed path between any pair of nodes).

WF-nets have been widely used since they have demonstrated to be a natural representation of BPM processes where each process has a well-defined beginning and end. In-between these, activities are performed according to a predefined workflow. Moreover, one model may be instantiated many times. In this respect, a process can be modeled as a WF-net were each instance is a copy of the same model [19].

However, the violation of some properties inherited from Petri nets (e.g., deadlock free, liveness [14]) may produce a model that is not correctly defined. For this reason, correctness criterion for WF-nets like soundness, have been defined [19]. A Marked WF-net (N,M) where i is the source place and o is the sink place is sound when it fulfill 4 requirements:

- Safe: $(N, [i])$ is safe. This is an inherited property from Petri nets, it ensures that no place ever holds more than 1 token for any firing sequence.
- Proper completion: for any possible marking M, if there is a token in the final place there should not be another token in the net.
- Option to complete: from any possible marking M there should be a firing sequence that leads to proper completion where only the final place has a token
- Absence of dead parts: $(N, [i])$ contains no dead transitions, i.e, for each transition of the net, there is a firing sequence enabling it.

As these properties hold on the models shown in Figure 1, these are all sound WF-nets.

2.2 Event Logs

Event logs are considered the cornerstone for the application of process mining techniques. As mentioned in the previous section, they may come from a variety of sources and with a different level of maturity. Techniques for log extraction are being studied by the community [20]. Table 1 shows a fragment of an event log where each row corresponds to an event. In order to apply process discovery techniques, each event must correspond to a process instance, it should be possible to infer the order of the events, and each event should be related to some activity [19]. These elements correspond to columns Case Id, Timestamp and Activity respectively. Additionally, logs may contain additional information that can be used by other types of process mining techniques.

Case Id	Event Id	Properties			
		Timestamp	Activity	Resource	Cost
1	2671	10-09-2014:08.30	ActivityA (a)	John, Peter	2
1	2672	11-09-2014:15.35	ActivityB (b)	John, Lucas	4
1	2673	13-09-2014:08.00	ActivityE (e)	Maria, Lucas	3
1	2674	15-09-2014:09.43	ActivityD (d)	Peter, John	1
2	2771	23-09-2014:08.30	ActivityA (a)	John, Carlos	3
2	2772	24-09-2014:09.33	ActivityE (e)	Peter	5
2	2773	27-09-2014:17.43	ActivityC (c)	Maria	2
2	2774	30-09-2014:10.18	ActivityD (d)	John, Carlos	1
...	...	...	...	...	

Table 1: Log L_1 - fragment showing two cases.

Since we do not need all event properties, we use a more succinct representation. Each event activity is represented with an identifier (a letter). Each type of case is represented as a sequence (also called trace) of these identifiers. Finally, the log is represented as a multiset of type of cases. For example, $L_1 = [\langle a, b, e, d\rangle^9, \langle a, e, c, d\rangle^8, \langle a, c, e, d\rangle^5]$ represents the full log shown in table 1 with 22 process instances (cases) and 3 types of sequences (traces). The standard format for storing logs is XES (eXtensible Event Stream) [26], which is a successor of MXML (Mining eXtensible Markup Language) [24], both are XML-based.

2.3 Process Discovery

Process discovery has been recognized as one of the most challenging PM tasks. Given an event log L (e.g., a XES log), a process discovery algorithm is a function that maps the log onto a process model (e.g., a WF-net) that is "representative" of the behavior seen in the event log. This representative model may exhibit different perspectives (e.g., control-flow, organizational or resource, data perspective) depending on what is needed and the data available in the log. In this survey, the focus is on control-flow since most of the discovery algorithms are used for this perspective (e.g., $\alpha - algorithm$ [19], *Heuristic miner* [29], *Genetic miner* [12]).

3 Quality Criteria

Soundness can be used to some extent to evaluate the correctness of a model. However, lets assume that we have log L_1 and that we have used three different process discovery algorithms, leading to the discovering of the models shown in Figure 1. As all these models are sound, there is a need for a more descriptive way of evaluating the quality of a model. To address this issue, 4 criteria have been defined [21] [19] [15]. Most of these measures are based on the replayability (also called parse) of the log traces on the model. Given a marked WF-net with a token in the initial place, the parsing of a trace is the firing of each of the events as if it were a firing sequence. If the trace results in proper completion, it is said to be properly parsed. If during the parsing, the current activity is

not enabled by the firing rule, the net is said to have missing tokens. Finally, if the parsing of a trace ends, but proper completion property does not hold, it is said to have remaining tokens.

3.1 Fitness

This evaluation criterion indicates how much of the observed behavior in the log is captured by the process model [15]. A discovered model with good fitness should allow the replay of (most of) the behavior seen in the event log [21]. For example, all the models shown in Figure 1 are a perfect fit for log L_1. However, lets say we add sequence $\sigma = a, b, c, b, e, d$ to L_1. Now, only M_3 is a perfect fit since the additional sequence can only be properly parsed by this model.

3.2 Simplicity

In order for a process model to be useful, it needs be interpreted and understood by a variety of people with different backgrounds. To make this easier, the discovered model should be as simple as possible. For example, there are often several syntactic ways to express the same behavior, and there may be preferred and less suitable representations [15]. With respect to this quality criterion, the simplest model that can explain the behavior contained in the log is the best model [21]. For example, we can say that M_2 is the most complex model between the models shown in Figure 1, as it has the highest number of places, transitions and edges.

3.3 Precision

Fitness and simplicity are not enough to judge the quality of a discovered process model [21]. Consider M_3: this model is known as a *flower model*, and can replay all traces in log L_1, but also any other log referring to the same set of activities. To meet this criterion, a model should fulfill certain precision property that focus on avoiding overly general models [15]. The discovered model should not allow for behavior very different from what was seen in the event log. For example, M_2 has perfect precision since the model does not allow for any behavior other than seen there in log L_1. M_1 has a good precision since it allows additional behavior not seen in the log (i.e., $\sigma = a, e, b, d$) but these behavior is similar to those seen in the log. Finally, M_3 has low precision, since it allows for an infinite number of traces that are not present in the log, most of which are not similar to the behavior represented in the log.

3.4 Generalization

As opposed to precision, this criterion focus on avoiding overly precise models [15]. Event logs are often far from complete. Consider a model where 10 activities can be executed in parallel. A log would have to contain at least $10! = 3{,}628{,}800$ traces to represent all possible behavior [21]. Since it is very unlikely

that all of this behavior is registered in a log, the model extracted from it should be general enoigh to include similar behavior that has not been registered in the log. For example, M_3 is very general since it allows many behaviors that are not registered in the log. On the other hand, M_2 is not general at all since it only models the behavior seen in the log. Finally, M_1 has a good level of generalization since it models similar behavior that is not present in the log.

4 Metrics

During the application of a process discovery algorithm we start with a log (L) and end up with a mined process model (M_m). Even though process discovery builds the model without an a-priori known model, in an experimental setting, we may assume that we know the original model (M_o) for evaluation purposes. It is possible to compare the discovered model M_m with the event log resulting in a *conformance checking* subproblem [16]. On the other hand, the mined M_m and original Mo model may be compared, which leads to the *process equivalence* problem [3]. Finally, some simplicity evaluation only take into account the mined model M_m.

The metrics discussed in this section are bounded to those used to evaluate the results of process discovery. This section describes those metrics. The formulas or steps needed to calculate these metrics are presented in Appendix A. A summary of the metrics are shown in Appendix B, along with the range of values reported in the evaluation of process discovery algorithms.

4.1 Parsing Measure [29] $PM(L, M_m)$

This is the most basic and coarse-grained function to evaluate fitness. It was defined to calculate the fitness of models generated using the Heuristic Miner [29] algorithm, returning a value between 0 and 1 indicating how much of the behavior in the log L is captured by M_m at a trace level. A high value indicates that most of the traces could be properly parsed without problems. A low value indicates that most of the traces exhibit missing or remaining tokens during the replay.

4.2 Partial Fitness Complete [12] $PF_{complete}(L, M_m)$

It measures the same as metric 4.1, but at a task level. i.e., it checks for the number of events (instead of complete traces) that could be parsed without problems during replay. This metric, together with *partial fitness precise* were first used to define a stop criterion for the genetic miner algorithm [12]. Similar to PM, it returns a value between 0 and 1 representing to the proportion of events that could be fired without problems during the replay.

4.3 Behavioral precision B_P and recall [12] $B_R(L, M_m, M_o)$

These metrics, along with the ones described in Section 4.4 were defined in order to evaluate the models obtained by the genetic miner [12] algorithm. All those metrics are based on replay of an event log by the mined and the original model. The B_P and B_R check for precision and generalization, respectively, based on the number of enabled transitions during the replay at a task level. As such, B_P checks how much behavior is allowed by the mined model that is not by the original model, and B_R checks for the opposite. Both return a value between 0 and 1. The more enabled tasks that the models have in common, the higher the values, meaning that the behavior represented in the model is more similar with respect to the event log.

4.4 Structural precision S_P and recall [12] $S_R(M_m, M_o)$

These metrics assess how many elements the mined and original model have in common, from a structural point of view. These metrics are computed over the causality relations from the models. The more causality relations that two models have in common, the more similar their structures are. S_P assesses how many causality relations the mined model has that are not in the original model. S_R checks the opposite, i.e., how many causality relations from the original model are not included in the mined model.

4.5 Extended Cardoso Metric [10] $ECaM(M_m)$

This metric, along with those described in Section 4.6 and 4.7, were defined to capture the complexity of models. The range of their values depends on the size of the model. The higher the value, the more complicated the model. Since these metrics are not normalized, they cannot be directly used for comparisons. Thus, they are excluded from the table in Appendix B. A normalized version of these metrics was used in [7] but the authors do not explain how they were normalized. $ECaM$ extends the Cardoso metric [4], which counts the various splits (XOR, OR, and AND) in the model and give each of them a certain penalty. This metric penalizes each place p by the number of subsets of places reachable from that place [10]. The higher the value, the more complex the model, since it has more splits.

4.6 Extended Cyclomatic Metric [10] $ECyM(M_m)$

This is an extension of the McCabe's Cyclomatic metric [11], which was intended to reveal complex pieces of imperative code by measuring the complexity of the control-flow associated graph. The $ECyM$ metric is similar measuring instead the complexity of the *reachability graph* [14] of a WF-net, as a way of measuring its complexity. Thus, this metric measures the model based on the elements of the reachability graph: number of edges, vertices and connected components. The higher the value, the more complex the model because it exhibits more potential behavior.

4.7 Structuredness Metric [10] $SM(M_m)$

This metric takes into account the limitations of metrics 4.5 and 4.6. $ECaM$ (4.5) only focuses on the syntax of the model and ignores the complexity of the behavior. $ECyM$ (4.6) focuses on the behavior and ignores the complexity of the model. SM defines 7 different component types in the model (e.g., sequence, choice, while, marked graph) and assigns a weight to each of them according to their complexity. The main idea is to identify the presence of these component types in the model and reduce them until the WF-net is trivial (i.e., with just one transition). A high value indicates that the model has complex components and therefore the model itself is more complex.

4.8 Fitness [16] $f(L, M_m)$

This metric, along with those discussed in Section 4.9 and 4.10, were first used for checking conformance between an event log and its corresponding model [16]. This metric measures the fitness of the model by replaying of every type of trace. As replay advances, it counts the number of tokens that had to be artificially added because the transition belonging to the logged event was not enabled and therefore could not be successfully fired, as well as the number of remaining tokens. This metric returns a value between 0 and 1. A high value indicates that few tokens had to be artificially added and that there are few remaining tokens after replay, meaning that the model is a good fit with respect to the model.

4.9 Advanced behavioral appropriateness [16] $a'_B(L, M_m)$

This metric is used to measure the *behavioral appropriateness* of a model, i.e., how much behavior is allowed by the model which is not present in the log. It focuses on the intuition that more precise process models are more desirable, since overly general models (like the flower model in Figure 1c) are less informative as they no longer describe the process registered in the log. a'_B compares the variability of the behavior allowed by the model and the behavior observed in the log, based on the cardinality of sometimes follows (S_F) and sometimes precedes (S_P) relations between each pair of activities. This metric returns a value between 0 and 1. The higher the value, the more similar the behavior of the model and the log are.

4.10 Advanced structural appropriateness [16] $a'_S(M_m)$

This metric is based on the fact that syntactically there are several ways to express the same behavior in a process model. However, the presence of some constructs may result in a more complex model: duplicate tasks (transitions with the same label), invisible tasks (transitions without a label or label τ) and implicit places (places that can be removed without changing the behavior of the model). This metric returns a value between 0 and 1. A higher value means

that less of the mentioned constructs appear in the model and it is therefore a simpler model.

4.11 Behavioral recall [8] r_B^p, specificity [8] s_B^n and precision [6] $p_B(L, M_m)$

These metrics are based on a technique that allows the injection of *artificial negative events* into an event log [8]. Negative events indicates that at a particular position in an event sequence cannot occur. The inclusion of this kind of events allows the use of metrics used in pattern recognition and information retrieval, through the construction of a confusion matrix [6]. The log (which contains so called positive events) is parsed with the supplemented negative events and gives values for the confusion matrix: true positives (enabled transition for positive event), true negatives (disabled transition for negative event), false positives (enabled transition for negative event), and false negatives (disabled transition for positive event). Based on these values, these metrics were defined for process discovery. r_B^p reflects how much behavior present in the event log is captured by the model. It assesses fitness, since a high value indicates that most of the behavior in the log is captured by the model. s_B^n checks how much of the introduced negative events are not present in the model. It gives a sense of precision, since a high value indicates that behavior that was not observed in the log is not included in the model. $p_B(L, M_m)$ indicates how much of the behavior (log and introduced negative events) are correctly shown in the model. All of these metrics return values in the [0,1] range.

4.12 ETC Precision [13] $etc_P(L, M_m)$

This metric was at first used for conformance checking. It checks for so called *escaping edges*, i.e., situations were the model allows for more behavior than the log. This metric returns a value between 0 and 1. A high value indicates the presence of escaping edges and therefore the model is less precise.

4.13 Structural similarity [28] (M_m, M_o)

This metric is based in the *dependency difference metric* (d) [2], which is used to compare two models. d returns the difference of the so called *process matrix* that indicates the direct follows relations between activities of the models to be compared. The structural similarity measure compares the mined and original model with a variant of d. This metric returns a value between 0 and 1. As its name implies, a high value indicates that the mined and original model are similar from a structural point of view.

4.14 Behavioral similarity [28] (M_m, M_o)

This metric is measured by the Principal Transition Sequence (PTS) similarity measure [27] of the *coverability tree* [14] from the mined and original models.

These transitions sequences indicate possible behaviors in the model. Behavioral similarity indicates the similarity of the transitions sequences of the models. A high value indicates the models are similar in behavior.

4.15 Precision [25] $P(L, M_m, M_o)$

This metric is a variant of the *precision* metric used in [22] for conformance checking which only takes into account the log and a model. The *precision* metric checks whether enabled activities in the model actually correspond to observed executions in the log. P checks the difference of the *precision* metric applied to the original and the mined model. The higher the difference, the less precise the mined model is.

4.16 Simplicity [25] $S(L, M_m, M_o)$

This metric is based on the *weighted average arc degree* defined in [17], which is the average of the number of both incoming and outgoing arcs calculated per node. *Simplicity* checks the difference between the weighted average arc degrees of the mined and original model. The higher the difference, the more complex the mined model.

5 Conclusion

The aim of this survey is to answer the research questions discussed in Section 1. With regards RQ1, Section 4 shows that there are several ways of measuring the quality of models mined with process discovery techniques. These metrics take as input the log L, the mined model M_m and when available, the original model M_o. The formulas or steps to calculate these metrics are listed in Appendix A. We can note that most of the metrics are normalized for values between 0 to 1. Few metrics require the original model. Precision is the criterion for which there are the most metrics in the literature. Regarding RQ2, Appendix B gives a summary of the discussed metrics along with the quality criteria they measure. The table in Appendix B also shows the range of reported values for 3 different types of logs: synthetic, real and synthetic logs with supplemented 5 to 10 % of noise. It should be noted that most of the papers included in this survey do not provide the logs that were used. This makes the comparison of the metrics values difficult. A noticeable exception is the work done in [25], where part of the evaluated logs are available online.

Appendix A Metric Formulas and Steps

A.1 Parsing Measure [29] $PM(L, M_m)$

$$\mathrm{PM}(L, M) = \frac{\mathrm{PPT}}{|T_L|} \tag{1}$$

$\mathrm{PPT}(L, M_m)$ # properly parsed traces from L in M.
T_L set of traces in L.

A.2 Partial Fitness Complete [12] $PF_{complete}(L, M_m)$

$$\mathrm{PF}_{complete}(L, M_m) = \frac{\mathrm{APA}(L, M_m) - punishment}{\mathrm{NAL}(L)} \tag{2}$$

$$punishment = \frac{\mathrm{AMT}(L, M_m)}{Traces(L) - \mathrm{TMT}(L, M_m) + 1} + \frac{\mathrm{AETLB}(L, M_m)}{Traces(L) - \mathrm{TETLB}(L,_M m) + 1} \tag{3}$$

$\mathrm{APA}(L, M_m)$ All Parsed Activities from L in M without problems.
$\mathrm{NAL}(L)$ Number of activities found in L.
$\mathrm{AMT}(L, M_m)$ All Missing Tokens during the parsing of all traces.
$\mathrm{AETLB}(L, M_m)$ All Extra Tokens Left Behind during the parsing plus the # tokens of end place minus 1 (proper completion).
$Traces(L)$ Traces in L.
$\mathrm{TMT}(L, M_m)$ Traces in which Missing Tokens were found during the parsing.
$\mathrm{TETLB}(L, M_m)$ Traces in which Extra Tokens were left behind during the parsing.

A.3 Behavioral precision B_P recall [12] $B_R(L, M_m, M_o)$

$$\mathrm{B_P}(L, M_o, M_m) = \frac{\sum_{\sigma \in L} \left(\frac{L(\sigma)}{|\sigma|} \times \sum_{i=1}^{|\sigma|} \frac{|Enabled(M_o, \sigma, i) \cap Enabled(M_m, \sigma, i)|}{|Enabled(M_m, \sigma, i)|} \right)}{\sum_{\sigma \in L} L(\sigma)} \tag{4}$$

$$\mathrm{B_R}(L, M_o, M_m) = \frac{\sum_{\sigma \in L} \left(\frac{L(\sigma)}{|\sigma|} \times \sum_{i=1}^{|\sigma|} \frac{|Enabled(M_o, \sigma, i) \cap Enabled(M_m, \sigma, i)|}{|Enabled(M_o, \sigma, i)|} \right)}{\sum_{\sigma \in L} L(\sigma)} \tag{5}$$

$Enabled(M, \sigma, i)$ # of enabled activities at M just before the parsing of the element at position i in trace σ.
$L(\sigma)$ # of traces σ in L.

A.4 Structural precision S_P and recall [12] $S_R(M_m, M_o)$

$$\mathrm{S_P}(M_o, M_m) = \frac{|\mathrm{C_o} \cap \mathrm{C_m}|}{|\mathrm{C_m}|} \tag{6}$$

$$\mathrm{S_R}(M_o, M_m) = \frac{|\mathrm{C_o} \cap \mathrm{C_m}|}{|\mathrm{C_o}|} \tag{7}$$

$\mathrm{C_x}$ causality relations from model M_x.

A.5 Extended Cardoso Metric [10] $ECaM(M_m)$

$$\mathrm{ECaM}(M_m) = \sum_{p \in P} \mathrm{ECFC_P}(p) \tag{8}$$

$$\mathrm{ECFC_P}(p) = |t \bullet | t \in p\bullet| \tag{9}$$

A.6 Extended Cyclomatic Metric [10] $ECyM(M_m)$

$$\mathrm{ECyM}(M_m) = |\mathrm{E}| - |\mathrm{V}| + p \tag{10}$$

E edges in the reachability graph of the model.
V vertices in the reachability graph of the model.
p # of strongly connected components in the reachability graph of the model.

A.7 Structuredness Metric [10] $SM(M_m)$

(i) $X := (M_m, \tau)$, where $\tau(t) = 1$, $\forall t \in T$.
(ii) while $|X| \neq \emptyset$
(iii) pick C so that $p_x(C) = min\{p_x(C')|C' \in |X|\}$
(iv) $M'_m := fold(M_m, C)$ where t_c is the added transition
(v) $\tau'(t_c) = \omega_X(C)$ and $\tau'(t) = \tau(t)$ for all other t
(vi) $X := (M'_m, \tau')$
(vii) Output $\mathrm{SM}(M_m) = \tau(t)(T = t$ after the net is reduced)

(i) X is the defined as a so called *Annotated WF-net* which is a WF-net with a function τ that assigns a weight to each transition in M_m. $\tau(t)$ starts in 1. (ii) The following steps are done until X is not trivial. (iii) In order to reduce the model it identifies components that match to predefined *component types*, namely: maximal sequence, choice, while, maximal marked graph, maximal state machine, maximal well structured and unstructured. Where each type has an associated value and weight. This step tries to find the one with the minimum value. (iv) the model is simplified by converting the identified component into a single transition. (v) τ is updated according to the weight of the identified component. (vi) X is reassigned with the new Annotated WF-net. (vii) finally τ of the last unique transition is returned as the value of $SM(M_m)$.

A.8 Fitness [16] $f(L, M_m)$

$$f(L, M_m) = \frac{1}{2}(1 - \frac{\sum_{i=1}^{k} \mathrm{n_i\, m_i}}{\sum_{i=1}^{k} \mathrm{n_i\, c_i}}) + \frac{1}{2}(1 - \frac{\sum_{i=1}^{k} \mathrm{n_i\, r_i}}{\sum_{i=1}^{k} \mathrm{n_i\, p_i}}) \tag{11}$$

k # of different traces.
$\mathrm{n_i}$ # of traces of type k in log L.
$\mathrm{m_i}$ # of missing tokens (artificially added) parsing i.
$\mathrm{r_i}$ # of remaining tokens parsing i.
$\mathrm{c_i}$ # of consumed tokens parsing i.
$\mathrm{p_i}$ # of produced tokens parsing i.

A.9 Advanced behavioral appropriateness [16] $a'_B(L, M_m)$

$$\mathrm{a'_B}(L, M_m) = (\frac{|\mathrm{S_F^l} \cap \mathrm{S_F^m}|}{2 \times |\mathrm{S_F^m}|} + \frac{|\mathrm{S_P^l} \cap \mathrm{S_P^m}|}{2 \times |\mathrm{S_P^m}|}) \tag{12}$$

S_F^l the set of tuples of activities with SF relation found in log L.
S_F^m the set of tuples of activities with SF relation found in model M_m.
S_P^l the set of tuples of activities with SP relation found in log L.
S_P^m the set of tuples of activities with SP relation found in model M_m.

A.10 Advanced structural appropriateness [16] $a'_S(M_m)$

$$\mathrm{a'_S}(L, M_m) = \frac{|\mathrm{T}| - (|\mathrm{T_{DA}}| + |\mathrm{T_{IR}}|)}{|\mathrm{T}|} \tag{13}$$

T set of transitions in M_m.
T_{DA} set of alternative duplicate tasks.
T_{IR} set of redundant invisible tasks.

A.11 Behavioral recall [8] r_B^p, specificity [8] s_B^n and precision [6]$p_B(L, M_m)$

$$\mathrm{r_B^p}(L, M_m) = \frac{\sum_{i=1}^{k} \mathrm{n_i\, TP_i}}{\sum_{i=1}^{k} \mathrm{n_i\, TP_i} + \sum_{i=1}^{k} \mathrm{n_i\, FN_i}} \tag{14}$$

$$\mathrm{p_B}(L, M_m) = \frac{\sum_{i=1}^{k} \mathrm{n_i\, TP_i}}{\sum_{i=1}^{k} \mathrm{n_i\, TP_i} + \sum_{i=1}^{k} \mathrm{n_i\, FP_i}} \tag{15}$$

$$\mathrm{s_B^n}(L, M_m) = \frac{\sum_{i=1}^{k} \mathrm{n_i\, TN_i}}{\sum_{i=1}^{k} \mathrm{n_i\, TN_i} + \sum_{i=1}^{k} \mathrm{n_i\, FP_i}} \tag{16}$$

k # of different traces.
n_i # of traces of type k in log L.
TP_i number of events that are correctly parsed for trace i.
FN_i number of events for which transition was forced to fire for trace i.
TN_i number of negative events for which no transition was enabled for trace i.
FP_i number of negative events for which a transition was enabled for trace i.

A.12 ETC Precision [13] $etc_P(L, M_m)$

$$\mathrm{etc_P}(L, M_m) = 1 - \frac{\sum_{i=1}^{|L|} \sum_{j=1}^{|\sigma_i|+1} |\mathrm{E_E}(s_j^i)|}{\sum_{i=1}^{|L|} \sum_{j=1}^{|\sigma_i|+1} |\mathrm{A_T}(s_j^i)|} \tag{17}$$

$E_E(s_j^i)$ Escaping edges at state s_j^i.
$A_T(s_j^i)$ Allowed tasks at state s_j^i.

A.13 Structural similarity [28](M_m, M_o)

$$Structural = \frac{1}{1+d} \tag{18}$$

$d(M_m, M_o)$ Dependency difference metric of M_m and M_o.

A.14 Behavioral similarity [28] (M_m, M_o)

(i) Construct the coverability tree for M_m and M_o.
(ii) Get the principal transition sequences from the trees.
(iii) Return the PTS-Based Similarity Measure based on the sequences found.

A.15 Precision [25] $P(L, M_m, M_o)$

$$\mathrm{P}(L, M_m, M_o) = 1 - max\{0, \mathrm{P'}_{L,M_o} - \mathrm{P'}_{L,M_m}\} \tag{19}$$

$$\mathrm{P'}(L, M) = \frac{1}{|\varepsilon|} \sum_{e \in \varepsilon} \frac{|\mathrm{en_L}(e)|}{|\mathrm{en_M}(e)|} \tag{20}$$

ε collection of *unique events* in a context of the log.
$\mathrm{en_M}(e)$ enabled activities in the model M.
$\mathrm{en_L}(e)$ observed activities actually executed in a similar context in L.

A.16 Simplicity [25] $S(L, M_m, M_o)$

$$\mathrm{S}(L, M_m, M_o) = \frac{1}{1 + max\{0, \mathrm{S'_{M_o}} - \mathrm{S'_{M_m}}\}} \tag{21}$$

$\mathrm{S'_M}$ the weighted average arc degree of the model M.

Appendix B Metrics Table

Metric	Artificial	Real	Artificial 5-10 noise	Fitness	Simplicity	Precision	Generalization
PM	[0 ; 1]	[0,015; 1]	[0 ; 1]	✓			
$PF_{complete}$	[0,993 ; 1]			✓			
B_P	[0,440 ; 1]		[0,050 ; 1]			✓	
B_R	[0,630 ; 1]		[0,050 ; 1]				✓
S_P	[0,080 ; 1]		[0,120 ; 1]			✓	
S_R	[0,710 ; 1]		[0,38 ; 1]				✓
f	[0,960 ; 1]	[0,523 ; 0,970]	[0,530 ; 0,990]	✓			
a'_B	[0,646 ; 1]	[0,433 ; 0,778]	[0,530 ; 0,890]			✓	
a'_S	[0,727 ; 1]				✓		
r_B^p	[0,952 ; 0,999]	[0,373 ; 0,960]	[0,380 ; 0,990]	✓			
s_B^n	[0,850 ; 0,987]	[0,62 ; 0,960]	[0,450 ; 0,960]			✓	
p_B	[0,862 ; 0,936]	[0,237 ; 0,516]					✓
etc_P	[0,902 ; 0,952]					✓	
$Structural$		[0,100 ; 1]				✓	✓
$Behavioral$		[0,750 ; 1]				✓	✓
P	[0,140 ; 1]		[0 ; 1]			✓	
S	[0,520 ; 1]		[0,070 ; 1]		✓		

References

[1] R. Agrawal, D. Gunopulos, and F. Leymann. Mining process models from workflow logs. In *Advances in Database Technology EDBT'98*, number 1377 in Lecture Notes in Computer Science, pages 467–483. Springer Berlin Heidelberg, 1998.

[2] J. Bae, L. Liu, J. Caverlee, L. Zhang, and H. Bae. Development of Distance Measures for Process Mining, Discovery and Integration:. *International Journal of Web Services Research*, 4(4):1–17, 2007.

[3] M. Becker and R. Laue. A comparative survey of business process similarity measures. *Computers in Industry*, 63(2):148–167, February 2012.

[4] J. Cardoso. Control-flow complexity Measurement of Processes and Weyuker's Properties. *Transactions on Engineering, Computing, and Technology*, 8:213–218, October 2005.

[5] J. E. Cook and A. L. Wolf. Discovering Models of Software Processes from Event-based Data. *ACM Trans. Softw. Eng. Methodol.*, 7(3):215–249, July 1998.

[6] J. De Weerdt, M. De Backer, J. Vanthienen, and B. Baesens. A robust F-measure for evaluating discovered process models. In *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 148–155, April 2011.

[7] 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*, 37(7):654–676, November 2012.

[8] S. Goedertier, D. Martens, J. Vanthienen, and B. Baesens. Robust Process Discovery with Artificial Negative Events. *J. Mach. Learn. Res.*, 10:1305–1340, June 2009.

[9] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman. Systematic literature reviews in software engineering A systematic literature review. *Information and Software Technology*, 51(1):7–15, January 2009.

[10] K. B. Lassen and W. M. P. Van der Aalst. Complexity metrics for Workflow nets. *Information and Software Technology*, 51(3):610–626, March 2009.

[11] T. J. McCabe. A Complexity Measure. *IEEE Transactions on Software Engineering*, SE-2(4):308–320, December 1976.

[12] A. K. A. de Medeiros, A. J. M. M. Weijters, and W. M. P. Van der Aalst. Genetic process mining: an experimental evaluation. *Data Mining and Knowledge Discovery*, 14(2):245–304, January 2007.

[13] J. Muñoz Gama and J. Carmona. A Fresh Look at Precision in Process Conformance. In *Business Process Management*, number 6336 in Lecture Notes in Computer Science, pages 211–226. Springer Berlin Heidelberg, 2010.

[14] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[15] A. Rozinat, A. K. A. de Medeiros, C. W. Günther, A. J. M. M. Weijters, and W. M. P. Van der Aalst. *Towards an evaluation framework for process mining algorithms*. Beta, Research School for Operations Management and Logistics, 2007.

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

[17] L. Sánchez-González, F. García, J. Mendling, F. Ruiz, and Mario Piattini. Prediction of Business Process Model Quality Based on Structural Metrics. In *Conceptual Modeling ER 2010*, number 6412 in Lecture Notes in Computer Science, pages 458–463. Springer Berlin Heidelberg, 2010.

[18] W. M. P. Van Der Aalst. The Application of Petri Nets to Workflow Management. *Journal of Circuits, Systems and Computers*, 08(01):21–66, February 1998.

[19] W. M. P. Van Der Aalst. *Process mining: discovery, conformance and enhancement of business processes*. Springer Science & Business Media, 2011.

[20] W. M. P. Van der Aalst. Extracting Event Data from Databases to Unleash Process Mining. In *BPM - Driving Innovation in a Digital World*, Management for Professionals, pages 105–128. Springer International Publishing, 2015.

[21] W. M. P. Van der Aalst, A. Adriansyah, A. K. A. de Medeiros, F. Arcieri, T. Baier, T. Blickle, J. C. Bose, P. Van den Brand, R. Brandtjen, J. Buijs, A. Burattin, J. Carmona, M. Castellanos, J. Claes, J. Cook, N. Costantini, F. Curbera, E. Damiani, M. de Leoni, P. Delias, B. F. Van Dongen, M. Dumas, S. Dustdar, D. Fahland, D. R. Ferreira, W. Gaaloul, F. Van Geffen, S. Goel, C. Günther, A. Guzzo, P. Harmon, A. ter Hofstede, J. Hoogland, J. E. Ingvaldsen, K. Kato, R. Kuhn, A. Kumar, M. La Rosa, F. Maggi, D. Malerba, R. S. Mans, A. Manuel, M. McCreesh, P. Mello, J. Mendling, M. Montali, H. R. Motahari-Nezhad, M. zur Muehlen, J. Munoz-Gama, L. Pontieri, J. Ribeiro, A. Rozinat, H. S. Prez, R. S. Prez, M. Seplveda, J. Sinur, P. Soffer, M. Song, A. Sperduti, G. Stilo, C. Stoel, K. Swenson, M. Talamo, W. Tan, C. Turner, J. Vanthienen, G. Varvaressos, E. Verbeek, M. Verdonk, R. Vigo, J. Wang, B. Weber, M. Weidlich, T. Weijters, L. Wen, M. Westergaard, and M. Wynn. Process Mining Manifesto. In *Business Process Management Workshops*, number 99 in Lecture Notes in Business Information Processing, pages 169–194. Springer Berlin Heidelberg, 2012.

[22] W. M. P. Van der Aalst, A. Adriansyah, and B. F. Van Dongen. Replaying History on Process Models for Conformance Checking and Performance Analysis. *Wiley Int. Rev. Data Min. and Knowl. Disc.*, 2(2):182–192, March 2012.

[23] B. F. Van Dongen, A. K. A. de Medeiros, and L. Wen. Process Mining: Overview and Outlook of Petri Net Discovery Algorithms. In *Transactions on Petri Nets and Other Models of Concurrency II*, number 5460 in Lecture Notes in Computer Science, pages 225–242. Springer Berlin Heidelberg, 2009.

[24] B. F. van Dongen and W. M. P. Van der Aalst. A Meta Model for Process Mining Data. *EMOI-INTEROP*, 160, 2005.

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

[26] H. M. W. Verbeek, J. C. A. M. Buijs, B F. Van Dongen, and W. M. P. Van der Aalst. XES, XESame, and ProM 6. In *Information Systems Evolution*, number 72 in Lecture Notes in Business Information Processing, pages 60–75. Springer Berlin Heidelberg, 2011.

[27] J. Wang, T. He, L. Wen, N. Wu, A. H. M. ter Hofstede, and J. Su. A Behavioral Similarity Measure between Labeled Petri Nets Based on Principal Transition Sequences. In *On the Move to Meaningful Internet Systems: OTM 2010*, number 6426 in Lecture Notes in Computer Science, pages 394–401. Springer Berlin Heidelberg, 2010.

[28] J. Wang, R. K. Wong, J. Ding, Q. Guo, and L. Wen. Efficient Selection of Process Mining Algorithms. *IEEE Transactions on Services Computing*, 6(4):484–496, October 2013.

[29] AJMM Weijters, Wil MP van Der Aalst, and AK Alves De Medeiros. Process mining with the heuristics miner-algorithm. *Technische Universiteit Eindhoven, Tech. Rep. WP*, 166:1–34, 2006.

[30] L. Wen, J. Wang, and J. Sun. Mining Invisible Tasks from Event Logs. In *Advances in Data and Web Management*, number 4505 in Lecture Notes in Computer Science, pages 358–365. Springer Berlin Heidelberg, 2007.

[31] C. Wohlin, P. Runeson, M. Höst, Magnus C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering*. Springer Science & Business Media, June 2012.