Imperative versus Declarative Process Mining
An Empirical Comparison
Back, Christoffer Olling; Debois, Søren; Slaats, Tijs

Publication date: 2020

Citation for published version (APA):
Imperative versus Declarative Process Mining: An Empirical Comparison

Christoffer Olling Back¹, Søren Debois², and Tijs Slaats¹ *

¹ Department of Computer Science, University of Copenhagen
Emil Holms Kanal 6, 2300 Copenhagen S, Denmark
{back, slaats}@di.ku.dk

² Department of Computer Science, IT University of Copenhagen
Rued Langgaards Vej 7, 2300 Copenhagen S, Denmark
{debois}@itu.dk

Abstract. Process modelling notations fall in two broad categories: declarative notations, which specify the rules governing a process; and imperative notations, which specify the flows admitted by a process. We study the question whether certain process logs are better suited for mining to imperative than declarative notations. We attack this question by applying a flagship imperative and declarative miner to a standard collection of process logs, then evaluate the quality of the output models wrt. the standard output model metrics of precision and generalisation. This approach requires perfect fitness of the output model, which substantially narrows the field of available miners; we use the Inductive Miner and Minerful. With the metrics in hand, we statistically evaluate the hypotheses that (1) one miner consistently outperforms the other on one of the metrics, and (2) there exist subsets of logs more suitable for imperative respectively declarative mining. We confirm both hypotheses: (1) declarative output models have statistically significant higher generalisation (with no significant difference on precision), and (2) we find subsets of logs for which the imperative respectively declarative miner significantly outperforms the other.

Keywords: Process Mining, Modelling Paradigms, Statistical Evaluation, Declarative Models, Imperative Models, Hybrid Models, Evaluation Metrics

1 Introduction

Workflow notations are commonly categorised as falling within either the imperative or declarative paradigm [19]. Imperative notations use flow-based constructs to explicitly model the paths through a process [1]. Declarative notations use constraint-based constructs to model the rules of a process. A declarative model allows all paths not forbidden by the constraints, and therefore the behaviour of the model is implicit in the rules and needs to be deduced by the system or users [11][12][24]. While the imperative paradigm is more mature, both paradigms have seen industrial adoption [17][18][21].

* This work is supported by the Hybrid Business Process Management Technologies project (DFF-6111-00337) funded by the Danish Council for Independent Research, and the EcoKnow project (7050-00034A) funded by the Innovation Foundation.
Regardless of notation, models have to come from someplace. A recent trend in both academia and industry is to extract models from real-life data via process discovery \cite{23,25}, where an output model is automatically constructed from an event log of observed process executions. Research into this approach has focused primarily on the discovery of imperative models, but substantial energy has been directed towards algorithms that discover declarative models as well \cite{6,10,14}.

Thus, when we construct process models by process discovery, we have a choice: Which paradigm should we use? Would we get better models from one than the other? Would such a difference be universal or would it depend on the particular input log? This paper studies empirically the ramifications of the choice of paradigm on output model quality \cite{4,5}. We specifically use the notation-agnostic metrics for precision and generalisation from \cite{27}, which apply equally to imperative and declarative models.

We use the Inductive Miner \cite{13} and MINERful \cite{6} as representatives of the imperative and declarative paradigms, respectively. We explain this choice in detail in Section \ref{sec:methods}. In short, fitness and simplicity are treated as independent variables: we require discovered models to have perfect fitness and “reasonable” simplicity. This limits our choice of miners, as at the time of writing the Inductive Miner is the only imperative miner guaranteeing perfect fitness \cite{13}, and MINERful is the only declarative miner that can be configured to produce a reasonable number of constraints for each of our input logs. Fortunately, the Inductive Miner and MINERful are widely considered to be at the cutting edge of their respective fields. Consequently, they make for reasonable representatives of their respective paradigms.

Evaluating these miners on the largest set of real-life logs available to us, we test the following hypotheses:

**Hypothesis 1**: One miner consistently outperforms the other on one of the metrics:

1a) one miner outperforms the other on precision.
1b) one miner outperforms the other on generalisation.

**Hypothesis 2**: There exists subsets of logs more suitable for imperative respectively declarative mining. Technically:

2a) There exists a subset of logs which when mined declaratively represent a Pareto improvement over the imperative miner; and this deviation from the zero mean lies outside of the bounds of what can be accounted for by random chance.
2b) There exists a subset of logs which when mined imperatively represent a Pareto improvement over the declarative miner; and this deviation from the zero mean lies outside of the bounds of what can be accounted for by random chance.

**Findings**

For Hypotheses 1, we find that MINERful outperforms IM on generalisation \((p < 0.05)\), but neither significantly outperforms the other on precision. For Hypotheses 2, we find that there are subsets more suitable for declarative mining and for imperative mining \((p < 0.05)\).

These results show that, within the constraints of our study, declarative miners can provide significantly more general models, without noticeably sacrificing precision. We
Imperative versus Declarative Process Mining: An Empirical Comparison

hereby provide a strong argument for the usefulness of declarative process discovery algorithms. In addition, we have identified one log that is significantly more suitable to imperative mining, and two logs that are significantly more suitable to declarative mining (see Table 4 for details). We believe that knowing which logs favour which paradigm will form a cornerstone for the development of hybrid mining algorithms that combine the strengths of each.

2 Related Work

Measures of \textit{output model quality}, in particular the four measures of fitness, precision, generalisation and simplicity \cite{4,5} are well-studied, e.g., \cite{20,23,27}. The particular variants of precision and generalisation used in the present paper originate with \cite{2,4,5}, as a basis of evaluating alignments for conformance checking. We employ the prefix automaton of \cite{16} to avoid state-space enumeration; prefix automata themselves arise as a certain choice of parameters for the framework of \cite{28}. Incidentally, precision and generalisation were originally formalised for Process Trees or Petri Nets, and thus did not originally apply to other models. It is precisely the later re-formulations in terms of prefix automata \cite{16} and labelled transition systems in \cite{27} that makes it it possible to make the present comparision of imperative and declarative output model quality.

The precision and generalisation metrics used in the present study both require \textit{perfect fitness}. There are two ways to achieve that: through cost-based alignment between log and model \cite{3,27} or replay techniques, e.g., \cite{20}; or by requiring outright that the miner produces only perfectly fitting models. To the best of our knowledge, besides the Inductive Miner and MINERful, only a small cadre of other miners provide perfect fitness, namely the Declare Maps miner \cite{14}, the DCR miner \cite{10}, and DISCO. Several other studies comparing miners exists. \cite{8} reported on a meticulous and comprehensive comparison in 2012. However, we are concerned only with miners that guarantee perfect fitness, and \textit{there were no such miners in 2012} \cite{13}. \cite{6} compares the declarative MINERful and Declare Maps miner, concluding that MINERful is much faster, but also very sensitive to its parameter settings. However, in all these studies, both aims and methods were quite different from the present paper: We study statistically the question whether there is sufficient evidence that declarative or imperative miners are generally or on some logs at an advantage; such a comparison, with or without statistical methods, has not been previously attempted. In \cite{9} the authors investigate the application of imperative process discovery to a process log originating from a declarative real-life process, and show that the original declarative model significantly outperforms the mined imperative models. However, the study is limited to a single real-life log and does not make a comparison between imperative and declarative miners.

The question “which notation?” is especially salient in the emerging field of \textit{hybrid mining} \cite{15,22} where output models comprise imperative and declarative sub-models.
3 Methods

We proceed as follows. (1) we select test data: a set of real-life event logs, (2) we run imperative and declarative miners on these logs, (3) we compute quality metrics on the output models, and (4) we evaluate statistically the hypotheses of Section 1.

3.1 Log Selection

For our test data we first gathered all of the real-life logs of the IEEE Task Force on Process Mining event log collection, the most commonly used publicly available event logs in process mining research. We added an additional real-life log from our own industrial contacts. We omitted the logs failing to fulfill the following criteria:

1. The logs must come in a supported file format. (E.g., we omitted CSV files that did not come with clear instructions on which columns represented event classes.)
2. Process discovery for the log must terminate and output a model for both the imperative and declarative mining algorithms.
3. The log must be an event log. (E.g., we omitted “logs” that did not represent a sequence of events.)

We report on included and omitted logs in Table 1. For logs associating lifecycle transition attributes with events, we incorporated the lifecycle transition into the name of the event to avoid, e.g., a start and complete event for an activity being interpreted as that activity occurring twice in the trace.

<table>
<thead>
<tr>
<th>EVALUATED</th>
<th>OMITTED</th>
<th>REASON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activities of Daily Living</td>
<td>Apache Commons Crypto Tests</td>
<td>Out of memory</td>
</tr>
<tr>
<td>BPI Challenge 2012</td>
<td>BPI Challenge 2014</td>
<td>Not well-formed event log</td>
</tr>
<tr>
<td>BPI Challenge 2013</td>
<td>BPI Challenge 2016</td>
<td>Not well-formed event log</td>
</tr>
<tr>
<td>BPI Challenge 2015 1-5</td>
<td>Credit Requirement Event Log</td>
<td>Deprecated format (MXML)</td>
</tr>
<tr>
<td>BPI Challenge 2017</td>
<td>Interactions w/ Lighting Interfaces</td>
<td>Not well-formed event log</td>
</tr>
<tr>
<td>BPI Challenge 2018</td>
<td>Hospital Log</td>
<td>MINERful out of memory</td>
</tr>
<tr>
<td>Dreyer Foundation</td>
<td>WABO/CoSeLoG 1-5</td>
<td>Error reading logs</td>
</tr>
<tr>
<td>Hospital Billing</td>
<td></td>
<td>programmatically</td>
</tr>
<tr>
<td>NASA CEV Event Log</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WABO/CoSeLoG - receipt phase</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Road Traffic Fine Management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sepsis Cases</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Overview of included and omitted logs

3.2 Process Discovery

We mined the selected logs both imperatively and declaratively, selecting miners according to the following criteria.

\[^1\] Retrieved from: [http://data.4tu.nl/repository/collection:event_logs](http://data.4tu.nl/repository/collection:event_logs)

[^4]: We used the “Bring Lifecycle to Event Name” plugin available in ProM, by Nick Tax
1. The miner must be configurable to always produce perfectly fitting models.
2. The miner must be configurable to produce models of reasonable simplicity.

The first criterion follows from both precision and generalisation requiring perfect fitness of the output model. It would have been an option to allow non-fitting output, and then use a model-log alignment \cite{2,27}, but without domain knowledge or access to an expert, we cannot know which exact alignment is more appropriate for the real-world log. This means that we would be evaluating not just the mining algorithm, but the combination of mining algorithm and alignment function. In particular, we would not know whether to attribute a result in favour of one miner over the other to the miner itself, or to an fortunate choice of alignment for that particular miner.

The second criterion follows from a tendency of declarative miners to produce output models containing hundreds of thousands of constraints. Such models are not practically useful: it does not matter how good precision and generalisation a model has, it is still unhelpful. To make sure we are comparing miners only at helpful model outputs, we limit our selection of declarative miners to those that allow us to regulate the maximum number of constraints in the output model.

The two criteria left us only two miners: The Inductive Miner and MINERful.

The Inductive Miner is an imperative miner developed by Leemans et al. \cite{13}. Arguably the premiere imperative miner in the field, it uses a divide-and-conquer approach to generate block-structured process models output as Petri nets.

We use version 6.8.415\footnote{Retrieved from: http://promtools.org/prom6/packages/InductiveMiner/} with stock settings, except that the noise threshold is set to 0.0 to ensure that the miner outputs perfectly fitting models. We note that the chosen version is the latest at the time of writing. For practical reasons, we run the IM outside the ProM environment, accessing it programmatically via its Java interface. To ensure stock settings, we initialise the miner with an unmodified MiningParametersIM() object; we have confirmed that these settings generates models identical to those output by the IM plugin in ProM 6.7.

MINERful is a declarative miner developed by Di Ciccio et al. \cite{6}. It uses a two-phase approach: in the first phase, a knowledge base of statistical information on the log is built; in the second, this knowledge is queried in order to infer the constraints of the process. The output is a Declare model, possibly including negative constraints.

MINERful has a configurable support threshold, an interest factor threshold, and a confidence threshold. When set to grant the miner maximum latitude, some output models have excessive number of constraints. By iteratively adjusting these settings until we find a model that has the highest possible number of constraints that does not exceed the number of edges in the imperative model, we ensure that the imperative and declarative models are of comparable simplicity. We note that there exists no widely accepted method for comparing the simplicity of imperative and declarative models. This leaves us with comparing the number of model elements as the most suitable approximation.
3.3 Computing Metrics

Defining standard measures for precision and generalisation remains an open research challenge because: (1) In the field of process mining there is a lack of negative input data, i.e. event logs typically only contain positive, and no negative examples. This means that the usual definition of precision used in data mining, which relies on having negative examples, can not be applied to process discovery. (2) The prevalence of unbounded loops in process models means that they often describe an infinite set of allowed behaviour. Therefore the intuitive definitions of precision and generalisation, which take into account all of the of behaviour allowed by the model, are not applicable in practice. Instead most metrics aim to reduce the measured behaviour of the model to a finite set of traces which are “most relevant”.

Metric Selection  To compare imperative and declarative models, we require metrics that can be applied to both equally. This means that they need to be defined on either the level of languages or transition systems. Accordingly we have chosen to employ the metrics introduced in [27], in particular:

*Precision* [27, p.10] measures the degree to which a model is “underfitting” or “allowing too much behaviour” relative to the input log. This particular metric is based on the notion of *escaping edges*, which represent a point at which the model allows behaviour not seen in the log. The measured amount of additional behaviour is kept finite by only considering the first divergent activity. i.e. an escaping edge may lead to a loop representing an infinite set of traces that did not occur in the log, but only the trace ending with the first divergent activity will be counted.

*Generalisation* [27, p.11] on the other hand measures the degree to which a model is “overfitting”: is there behaviour not allowed by the model and not exhibited in the log, but that can be reasonably expected to occur in the future? This particular metric approximates generalisation by estimating for each state in the model the likelihood that a new, hitherto unseen, activity will occur. This is estimation is based on the number of activities that have been observed, and how often the state was visited. Two alternatives are offered: *event-based generalisation* takes into account the number of visits to a state, *state-based generalisation* does not.

Implementation  Although ProM contains a plugin for computing the metrics of [27] on Petri nets, it does not offer support for declarative models. We also found it more convenient to run our tests as a batch-process where we could easily pipeline several operations (mining, metrics computation, analysis) on a set of multiple logs. Therefore we developed our own evaluation framework. The framework also makes our methods and results straightforward to reproduce. One can inspect the code and run it on our, or their own input data. We have extensively verified our framework, in particular by testing it on the examples and results reported in [27].

---

6 Available at: [https://bitbucket.org/coback/qmpm](https://bitbucket.org/coback/qmpm)
3.4 Statistical analysis

Formally, we consider the included logs listed in Table 1 a representative sample of all possible logs, and study the four random variables arising as the combination of one miner and one metric: precision of the output model of MINERful ($P_m$) and of the Inductive Miner ($P_i$); and generalisation of the output model of MINERful ($G_m$) and of the Inductive Miner ($G_i$). Define random variables $\delta_P$, $\delta_G$ as the relative performance of the two miners on precision and generalisation, respectively, as follows. We consider two alternative definitions, both of which we assume to be normally distributed: the difference and the logarithm of the ratio.

The ratio in the latter is intended to reflect those cases in which one miner performs proportionally much better despite a moderate absolute difference and vice versa, while the logarithm ensures that values will fall in the range $[-1, 1]$ as well as diminishing the values of extreme outliers. We will use the difference based formulation as the basis for hypothesis testing, while observing that both representations result in a similar distribution of the data.

<table>
<thead>
<tr>
<th>Difference</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta_P$</td>
<td>$P_i - P_m$</td>
</tr>
<tr>
<td>$\delta_G$</td>
<td>$G_i - G_m$</td>
</tr>
</tbody>
</table>

**Hypothesis 1** We test against the null hypothesis $H_0$ that the mean difference in performance between miners is zero, and that any observed deviation can be accounted for by random variation. Write $H^P_0 : \mu_P = \mu_0$ and $H^G_0 : \mu_G = \mu_0$ let $N$ denote the sample size, and assume our metrics independent:

$$
\mu_P = \frac{1}{N} \sum \delta_P \quad \mu_G = \frac{1}{N} \sum \delta_G \quad \mu_0 = 0
$$

We use a one-sample Student’s $t$-test to determine whether $\mu_0$ lies far enough outside the sample distribution to reject $H^P_0$ or $H^G_0$. The direction of the deviation will indicate which miner outperforms:

<table>
<thead>
<tr>
<th>Null hypothesis</th>
<th>$\mu_P &gt; 0$</th>
<th>$\mu_P &lt; 0$</th>
<th>$\mu_G &gt; 0$</th>
<th>$\mu_G &lt; 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H^P_0$ : $\mu_P = \mu_0$</td>
<td>IM</td>
<td>MINERful</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$H^G_0$ : $\mu_G = \mu_0$</td>
<td>-</td>
<td>-</td>
<td>IM</td>
<td>MINERful</td>
</tr>
</tbody>
</table>

**Hypothesis 2** Consider the 2-dimensional space arising from the linear combination of $\delta_P$ and $\delta_G$, and define the vectors $\delta$, $\mu$, $\mu_0 \in \mathbb{R}^2$, where:

$$
\delta = \begin{bmatrix} \delta_P \\ \delta_G \end{bmatrix} \quad \mu = \begin{bmatrix} \mu_P \\ \mu_G \end{bmatrix} \quad \mu_0 = 0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}
$$

We then group samples based on the signs of $\delta_P$, $\delta_G$: 
The proper statistical test in this case is the multivariate version of the t-test, namely the one-sample Hotelling’s $T^2$ with the null hypothesis that the true mean lies at the origin, i.e. $H_0 : \mu = \mu_0$ and that samples deviating from the mean will fall within a normal distribution around $\mu_0$.

This leads to two criteria which must both be fulfilled for the null hypothesis to be rejected: there must exist one or more logs which lie in either the declarative or imperative group and these logs must be far enough from $\mu_0$ such that these samples are unlikely to have been drawn from the null hypothesis distribution.

Logs in the “trade-off” category cannot contribute to rejecting the null hypothesis despite deviating significantly, since they do not represent a Pareto improvement and therefore cannot be said to be better suited to one mining paradigm.

4 Results

We report the results of running the Inductive Miner and MINERful on the included logs of Table 1 in Table 2. The relative performance of the two miners on the two metrics is plotted in Figure 2.

Hypothesis 1

We reject the null hypothesis $H_{G0}$ corresponding to hypothesis 1b, with the difference in favour of MINERful. We note that we reject the null hypothesis at both difference and ratio formulations of $\delta_G$, and even using a Bonferroni correction to account for our testing of two hypotheses ($p < 0.025 = 0.025$). We are unable to reject null hypothesis $H_{P0}$ corresponding to hypothesis 1a.

These results indicates (i) that MINERful does outperform the Inductive Miner wrt. generalisation, but (ii) neither miner outperforms the other on precision.

Hypothesis 2

In order to reject null hypothesis $H_0$ corresponding to hypothesis 2, we must show that there exist data points in quadrants I or III which lie far enough from the origin ($\mu_0$) as to have a very low probably of being randomly sampled from the distribution around $\mu_0$. With at least three logs satisfying these criteria, we are able to reject the null hypothesis.

From the formulation of $H_0$, we have the mean of the null hypothesis’ probability distribution, that is $\mu = \mu_0$. For the covariance matrix of this distribution, we only have

<table>
<thead>
<tr>
<th>$\text{sgn}(\delta_P)$</th>
<th>$\text{sgn}(\delta_G)$</th>
<th>Group</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>declarative</td>
<td>MINERful outperforms IM on both metrics</td>
</tr>
<tr>
<td>+</td>
<td>+</td>
<td>imperative</td>
<td>IM outperforms MINERful on both metrics</td>
</tr>
<tr>
<td>+</td>
<td>-</td>
<td>trade-off</td>
<td>IM achieves higher precision, but lower generalisation</td>
</tr>
<tr>
<td>-</td>
<td>+</td>
<td>trade-off</td>
<td>MINERful achieves higher precision, lower generalisation</td>
</tr>
</tbody>
</table>
## RAW METRICS

<table>
<thead>
<tr>
<th>Log</th>
<th>INDUCTIVE MINER</th>
<th>MINERFUL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Generalisation</td>
</tr>
<tr>
<td>BPIC 2012</td>
<td>0.2401</td>
<td>1.0</td>
</tr>
<tr>
<td>BPIC 2013</td>
<td>0.4044</td>
<td>1.0</td>
</tr>
<tr>
<td>BPIC 2015-1</td>
<td>0.0056</td>
<td>0.9993</td>
</tr>
<tr>
<td>BPIC 2015-2</td>
<td>0.0049</td>
<td>0.9997</td>
</tr>
<tr>
<td>BPIC 2015-3</td>
<td>0.0058</td>
<td>0.9998</td>
</tr>
<tr>
<td>BPIC 2015-4</td>
<td>0.0284</td>
<td>0.9994</td>
</tr>
<tr>
<td>BPIC 2015-5</td>
<td>0.0241</td>
<td>0.9997</td>
</tr>
<tr>
<td>BPIC 2017</td>
<td>0.0601</td>
<td>1.0</td>
</tr>
<tr>
<td>BPIC 2018</td>
<td>0.0633</td>
<td>1.0</td>
</tr>
<tr>
<td>Dreyer</td>
<td>0.2235</td>
<td>0.9988</td>
</tr>
<tr>
<td>H. Billing</td>
<td>0.6034</td>
<td>1.0</td>
</tr>
<tr>
<td>NASA CEV</td>
<td>0.695</td>
<td>0.9999</td>
</tr>
<tr>
<td>Production</td>
<td>0.0218</td>
<td>0.9966</td>
</tr>
<tr>
<td>WABO</td>
<td>0.363</td>
<td>0.9995</td>
</tr>
<tr>
<td>Road Fines</td>
<td>0.8181</td>
<td>1.0</td>
</tr>
<tr>
<td>Sepsis Cases</td>
<td>0.1905</td>
<td>0.9999</td>
</tr>
</tbody>
</table>

### Table 2: Precision, generalisation and attributes of output models. The bottom portion shows the difference and the logarithm of the ratio of metrics for models produced for a given log.
the current data sample to use as a basis for estimation. Using each point’s distance from \( \mu_0 \), we arrive at the covariance matrix \( \Sigma_0 = \begin{bmatrix} 0.017 & 0.004 \\ 0.004 & 0.029 \end{bmatrix} \). We then apply Hotelling’s \( T^2 \) to each data point to determine the probability that it was drawn from the distribution associated with \( H_0 \).

Since we have multiple logs which we will test separately, we must adjust the p-value threshold to reflect this. This is due to our formulation of hypothesis 2, which requires that just one log fall in quadrant I or III and lie significantly far from the mean. Without a correction, the chance of seeing an outcome with a probability \( p = 0.05 \) after 13 observations is \( 13 \times 0.05 = 0.65 \); more likely than not. Therefore, we use the adjusted p-value threshold \( p < \alpha_{\text{adj}} = 0.05/13 = 0.0038 \).

Using this approach, we find statistically significant evidence that some logs are better suited for imperative mining, some better suited for declarative mining. In particular, the log “NASA CEV” (quadrant I of Table 2) is in this sense an “imperative log”; whereas the logs “Activities”, and “BPI 2017” (quadrant III of Table 4) are declarative. The difference in performance of the two miners on these logs is drastic enough that it is unlikely to be due to factors other than the mining approach.

\[ \text{It might be argued that we are only testing the six logs in quadrants I and III, not all 13. In this case } p < \frac{\alpha}{6} = \frac{0.05}{6} = 0.0083, \text{ but this results in the same relevant logs achieving significance.} \]
Fig. 2: Left: Difference between output models for each metric with boxes representing interquartile range. Graphic for ratios omitted (not significant). Some outliers not displayed. Middle, right: Bivariate representation of difference, and logarithm of ratio of metrics for output models. Positive values represent an advantage to Inductive Miner, negative values an advantage to MINERful. Quadrants II and IV represent a trade-off.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Mean (µ)</th>
<th>Lower</th>
<th>Upper</th>
<th>p-value</th>
<th>Mean (µ)</th>
<th>Lower</th>
<th>Upper</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>δP</td>
<td>0.0286</td>
<td>-0.0485</td>
<td>0.1058</td>
<td>0.4344</td>
<td>0.0324</td>
<td>-0.1574</td>
<td>0.2223</td>
<td>0.7141</td>
</tr>
<tr>
<td>δG (event)</td>
<td>0.1269</td>
<td>-0.0308</td>
<td>0.2846</td>
<td>0.1051</td>
<td>0.1166</td>
<td>-0.0679</td>
<td>0.3011</td>
<td>0.1917</td>
</tr>
<tr>
<td>δG (state)</td>
<td>-0.1112</td>
<td>-0.1862</td>
<td>-0.0362</td>
<td>0.0072</td>
<td>-0.0578</td>
<td>-0.1033</td>
<td>-0.0124</td>
<td>0.0172</td>
</tr>
</tbody>
</table>

Table 3: Test of hypothesis 1 using Student’s one sample t-test, two-tailed.

5 Discussion

Our findings for Hypothesis 1 show that declarative miners are able to provide significantly more general models than imperative miners. This finding corresponds with the common understanding that declarative models are useful for describing flexible processes that exhibit a high degree of variability through special cases. It is surprising however that this improvement on generalisation can be achieved without significantly sacrificing precision. A common intuitive understanding of precision and generalisation is that they are each other’s inverse: precision aims at avoiding under-fitting the data, whereas generalisation aims to avoid over-fitting the data. Based on our study it would appear that, at least for the particular metrics used here, declarative miners manage to find a sweet spot where they allow reasonable deviations of the process, without allowing too much behaviour. On the other hand, if one insists that generalisation and
Table 4: Test of hypothesis 2 using multiple one-sample Hotelling’s $T^2$ tests. The sample is used to estimate the covariance matrix $\Sigma_0 = \begin{bmatrix} 0.017 & 0.004 \\ 0.004 & 0.029 \end{bmatrix}$ of the distribution of the null hypothesis with mean at the origin ($\mu_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$). Then each log is tested to determine whether it lies significantly far from the origin. Logs with $p < \alpha/n = 0.0038$ are unlikely to be sampled from the distribution associated with the null hypothesis.

6 Conclusion

In this study, we systematically compare the performance of imperative versus declarative process mining algorithms based on the commonly accepted quality metrics for precision and generalisation defined by [27]. We investigate two hypotheses: first, that one miner performs better on a) precision and/or b) generalisation; second, that there exist some logs on which either miner provides a statistically significant Pareto improvement on both precision and generalisation. We find that declarative discovery provides a statistically significant higher generalisation on average, without sacrificing on precision should be negatively correlated, then our study underlines the need to carefully consider the interplay between the different quality dimensions when developing specific metrics.

Our findings for Hypothesis 2 identify at least three logs which are significantly more suited for either declarative or imperative mining, showing that there are particular circumstances in which each paradigm clearly outshines the other. Other logs either required a trade-off between precision and generalisation, or the difference between the two miners could not be considered statistically significant. It should also be noted that for one log, “Hospital Event Log”, MINERful was unable to return a model, so by default this log falls to Inductive Miner’s favour.
precision. We are also find that at least 3 out of 13 tested logs are clearly better suited to one mining paradigm over the other.

We recognise some threats to the validity of the study: the size of our sample set, the particular metrics used and the number of miners considered. Our choices and reasons are discussed in detail in Section 3. Each threat can be broken down to the current maturity of the field: there exist only a small number of publicly available real-life logs, there are only few metrics that can be equally applied to imperative and declarative models, and these metrics in turn limit our choice of miners. We therefore posit that the limitations placed on the study are reasonable given the state of the art. In addition, to the best of our knowledge, this has been the most exhaustive study comparing imperative and declarative process discovery techniques to date.

By showing that declarative mining algorithms are able to find more general models on average, we provide a strong argument for their usefulness. In addition, the identification of logs that are significantly more suitable to one paradigm over the other shows that there are real-life scenarios for which one paradigm outperforms the other. We expect that this study will open the door to a more thorough investigation of the properties that make a log more suitable to a paradigm, which in turn will drive the development of hybrid process discovery approaches that combine both paradigms.

References


