Exploring Processes and Deviations

Sander J.J. Leemans, Dirk Fahland, and Wil M.P. van der Aalst

Eindhoven University of Technology, the Netherlands
{s.j.j.leemans, d.fahland, w.m.p.v.d.aalst}@tue.nl

Abstract In process mining, one of the main challenges is to discover a process model, while balancing several quality criteria. This often requires repeatedly setting parameters, discovering a map and evaluating it, which we refer to as *process exploration*. Commercial process mining tools like Disco, Perceptive and Celonis are easy to use and have many features, such as log animation, immediate parameter feedback and extensive filtering options, but the resulting maps usually have no executable semantics and due to this, deviations cannot be analysed accurately. Most more academically oriented approaches (e.g., the numerous process discovery approaches supported by ProM) use maps having executable semantics (models), but are often slow, make unrealistic assumptions about the underlying process, or do not provide features like animation and seamless zooming. In this paper, we identify four aspects that are crucial for process exploration: *zoomability*, *evaluation*, *semantics*, and *speed*. We compare existing commercial tools and academic workflows using these aspects, and introduce a new tool, that aims to combine the best of both worlds. A feature comparison and a case study show that our tool bridges the gap between commercial and academic tools.

Keywords: process exploration, multi-perspective process mining, process deviation visualisation, conformance analysis

1 Introduction

Process mining, and in particular process discovery, have gained traction as a technique for analysing actual process executions from event data recorded in event logs. Process mining is typically used to learn whether, where, and how a process deviated from the intended behaviour. However, such information is usually not obtained by just running a single algorithm on an event log. A wide variety of (combinations of) algorithms can be used [20,19,18,2,14], typically heavily relying on various parameter settings to reveal and analyse specific aspects and features of a process, depending on the specific interests of the process stakeholder. Here we coin the term *process exploration* which refers to repeated parameter selection and tuning, iteratively performing process discovery, and continuously evaluating the resulting process map [1].

Interestingly, academic and commercial process mining tools support different aspects of process exploration. In this paper, we demonstrate that process exploration can be improved by combining beneficial features of academic and commercial tools. Existing commercial and some academic tools for process exploration, such as the Fuzzy Miner (FM) [11], Fluxicon Disco (FD) [12], Celonis Discovery (CD) and Perceptive

Process Mining (PM), are based on showing directly-follows graphs: nodes are process steps, and in general edges mean that an activity followed another. Thus, one can inspect the process by considering the arrows between them. These visualisations are intuitive, an example is shown in Figure 1, and some tools allow for extensive log filtering. Although directly-follows graph-based maps are useful for global analysis, they have some limitations. For instance, pure directly-follows based maps do not show parallelism, implying that in a map, the state of the system unrealistically solely depends on the last executed process step. Some tools support parallelism by sacrificing executable semantics, but their maps *cannot be used for automated analysis*; the maps do not show crucial features (e.g., types of splits and joins) and it is impossible to reason over them (e.g., which traces are possible). (In this paper, we refer to a process map with executable semantics as a process *model*.)

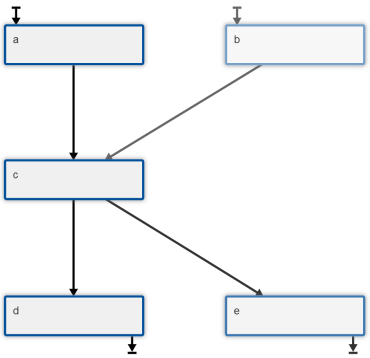


Figure 1: Example of a process map, discovered by Perceptive.

To evaluate a model with respect to its event log, several established quality metrics exist, for instance fitness, precision and generalisation [7]. Fitness describes what part of the log is expressed by the model, precision what part of the model is present in the log, and generalisation what part of future behaviour will be expressible by the model. While useful for model comparison, these measures are coarse grained: they provide a number for a model. Using more fine-grained measures and visualisations, *a detailed analysis of where the model deviates from the log and where other problems occur can be performed*. These evaluations require executable semantics and certain guarantees, for instance that the model contains no deadlocks, livelocks or other anomalies: that the model is *sound*. On an unsound model, for instance, only an upper bound of fitness can be computed reliably [13]. Only if a discovered process map is sound and has executable semantics, its quality with respect to the event log can be assessed accurately.

Academic tools are usually focused, powerful and the maps produced by them usually have executable semantics. Therefore, deviations and quality metrics can be studied. However, academic tools or plug-ins are often designed for one particular purpose, and combining tools, if possible, challenges usability. The ProM framework [9] streamlines cooperation between tools, as input and output formats of plug-ins of the framework are standardised. However, consider for instance a typical process exploration workflow in the ProM framework: to mine a model from a log using ILP miner and assessing the quality of the model, one has to click through 10 pop-up screens of parameters and options.

In this paper, we aim to bridge this gap between commercial and academic tools by introducing a process exploration tool, *Inductive visual Miner* (IvM), that provides the features of commercial tools and aims to be as user-friendly, while providing maps with semantics and built-in deviation visualisation. We consider some desirable features from both commercial and academic tools, and describe how IvM improves on them.

A prototype of IvM has been implemented as a plug-in of the ProM framework and is available for download from http://promtools.org. We perform a feature comparison and a case study on real-life logs.

The remainder of this paper is organised as follows: we first provide some background on process exploration. In Section 3 we analyse existing process exploration tools and discuss design decisions for IvM. A high-level feature comparison and a case study are performed in Section 4; Section 5 concludes the paper.

2 Process Exploration

Process exploration enables users to learn information from an event log. In this section, we first give an example of a process exploration case study, after which four aspects of process exploration are explored: zoomability, evaluation, speed and semantics.

Example. We illustrate process exploration using the winning case study of BPIC12 [4]. In this case study, first a high-level overview was generated to get an initial idea of the complexity of the given event log. The complexity was reduced by applying activity and life cycle filters to the event log. Next, a filter leaving only successful traces was applied and a high-level map was created, showing the 'happy flow' through the process, i.e. the path taken by an average trace. On this happy flow, business impact was measured, i.e. how many traces were successful or rejected with respect to each activity in the happy flow, leading for instance to the conclusion that "Nearly 23% of the applications that go to validation stage are declined, indicating possibilities for tightening upfront scrutiny at application or offer stage". Further on in the case study, an analysis was made whether the outcome of a trace (successful or rejected) was predictable during execution, which for instance led to the conclusion that "slow moving applications had a less than 6% chance of getting to approval". The authors note that this analysis, which was performed using decision tree miners, could be repeated at other stages.

This single case study already clearly shows the repeated process of setting parameters, selecting filters, generating process maps and continuously evaluating the results.

Zoomability. In the case study of [4], the log was examined on a high level, and then repeatedly examined in detail for different perspectives, e.g. by applying filters and using both high-level and detailed process maps. Compare it to electronic road maps: users can get a high-level view to see highways, or can zoom in to see alleys. Moreover, different perspectives can be shown, such as bicycle or public transport maps. A process exploration tool should support similar features by enabling a user adjust the level of detail in a process map (e.g. highways and alleys) and to filter it in several ways (e.g. bycicle maps); we refer to this ability as *zoomability*.

As the case study shows, a plethora of filtering options must be available: filters on event name (prefix), frequency, redundancy, data attributes and on resources were all used. Moreover, as used, the tool should be able to provide both a process map showing only the frequent paths of the process, as well as one with the outliers; i.e. maps with several levels of noise filtering. Many more filters are imaginable, however giving an exhaustive list of them is outside the scope of this paper.

Another powerful zoomability parameter is time: using log animation, a user can inspect this time perspective: the event log is visually replayed on the map, which re-

veals frequent paths and bottlenecks over time, and makes concept drift explicit. If an animation can be paused, it gives a frozen view of the map with the traces that were in the process at a particular point in time.

Evaluation. Given that the quality criteria fitness, precision, generalisation and simplicity compete [7], a perfect model often does not exist. Any process discovery algorithm has to make a trade-off between these criteria, so there may be many Pareto optimal models without there being a clear "best" one. For instance, low fitness could indicate a high-level model that is well-suited for getting the idea, but ill-suited for drawing high-confidence detailed conclusions. High precision indicates that the model closely resembles the behaviour of the event log, while a low precision indicates that the model allows for much behaviour that never happened. A model with bad generalisation has little predictive value (overfits), as it only describes the behaviour of the event log. More specifically, for instance the question whether a violation to the four-eyes principle occurred, i.e. whether two different persons looked at a certain case, should not be answered using a model with 80% fitness, as roughly 20% of the behaviour is not shown in it. Another example: a question whether something *could* happen in the future should be answered on a model with high generalisation, rather than an overfitting one.

Thus, conclusions based on discovered models should be drawn carefully while accounting for the quality criteria, and in order to find the best model to answer the question at hand, a process exploration tool should enable a user to evaluate a map.

A quality measure is typically expressed as a number, e.g., a fitness of 0.8. However, a single number for a complete model might be less informative. For example, one half of the process model could have a fitness of 0.6, while the other half might have a fitness of 1.0. A process exploration tool should provide detailed quality indicators, to locate problems in specific parts of the model.

Speed. The speed of process exploration is determined by two elements: the learning curve for users and the responsivity of the tool. Many aspects influence the learning curve of a tool; we highlight two: the map should use a representation that is easy to read by people who are not computer scientists, and the learning curve can be more gradual if the tool invites users to play with its parameters (for which the tool should be responsive).

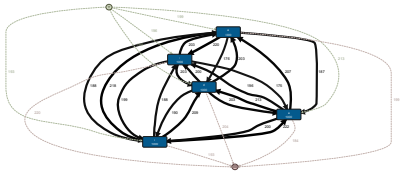
Responsivity is a challenge that comes with zoomability; exploration requires interaction: a user should neither have to wait long nor have to perform many actions to adjust the zoom; understanding a process requires a quick and responsive user interface.

Semantics & Guarantees. As described in the introduction, executable semantics and guarantees are essential for evaluation in a process exploration tool. Executable semantics allow for replay, which enables decision point analysis (which was performed in the BPIC12 case study: for a specific point in the process it was analysed what made a trace likely to succeed, and the authors note that they would like to repeat this experiment for other points in the process; using a model, the decision points would be known, allowing for automation [15]), enables prediction [22], and enables compliance checking [16].

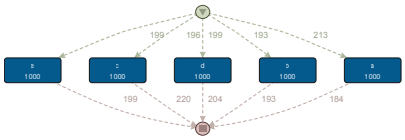
3 Existing Tools and Design Decisions

In this section, we analyse existing tools with respect to the requirements discussed in Section 2. Meanwhile, we describe the design decisions we made for the Inductive visual Miner (IvM).

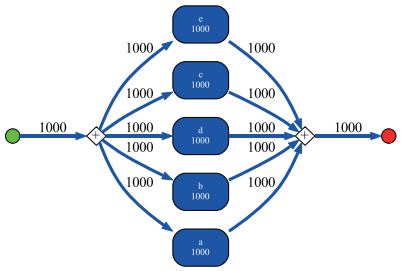
Tools. In this paper, we consider the following commercial tools: Fluxicon Disco (*FD*)[1] [12], Celonis Discovery (*CD*)[2] and Perceptive Process Mining (*PM*)[3]. For the academic tools, we consider three chains of plug-ins within the ProM framework: 1) Fuzzy Miner [11] (*FM*), 2) the chain (*IMi-C*), consisting of Inductive Miner - infrequent (IMi) [13], followed by PNetReplayer [3] and Project Manifest to Model for Conformance, and 3) the chain (*ILP-C*), consisting of ILP miner (ILP) [21], followed by PNetReplayer [3] and Project Manifest to Model for Conformance. We need to consider chains of plug-ins to allow for a fair comparison.



(a) Parallelism clutter.



(b) FD. Ambiguous.



(c) IvM.

Figure 2: Examples of tools applied to a log containing five parallel activities.

Representation and Process Discovery Technique. The first choice to make for a process exploration tool is what discovery technique and which representation to use.

The existing tools use three categories of discovery techniques: directly-follows based (FD, CD, PM, FM)[4], inductive mining (IMi) and optimisation problem mining (ILP). Directly-follows based tools either provide no semantics (FD, CD, FM) or do not support parallelism (PM), but are fast and allow for filtering; ILP provides semantics, and guarantees perfect fitness and best-possible precision, but does not guarantee soundness; IMi strikes a balance: it is fast, guarantees soundness, can guarantee fitness, and allows for noise filtering.

The learning curve of a tool is important for the speed aspect. We consider the directly-follows based representations to have the most gradual learning curve, and the Petri net based representations to have the steepest. Therefore, in order to obtain the most gradual learning curve, we design our representation to be as close as possible to

[1] http://fluxicon.com/disco/; April/May 2014

[2] http://www.celonis.de/en/discover/our-product; April/May 2014

[3] http://www.perceptivesoftware.co.uk/products/perceptive-process/process-mining; fast miner, April/May 2014

[4] Some tools (FM) can also take the eventually-follows relation into account.

the directly-follows based representation, but we add parallelism while keeping semantics.

For logs containing parallelism, directly-follows based tools usually connect all parallel activities, which yields clutter. For instance, the parallel execution of 5 activities yields a clique containing 20 edges (Figure 2a). A strategy to reduce this clutter is to manually filter out the parallel activities, as done in the BPIC12 case study [4]. Another strategy, used in for instance FD and FM, is to filter these edges (Figure 2b). However, Figure 2b looks exactly like the exclusive choice between 5 activities; only the numbers on the edges, denoting the frequency with which an edge was taken, tell the difference. So, while fixing parallelism, ambiguity is introduced.

In IvM, behind the scenes we use a variation of IMi. Internally, IMi and IvM use so-called process trees to ensure sound models. However, the results are shown to the user using a directly-follows based representation to stay close to that representation and its learning curve; we extend it with a start state, an end state and Petri net places to provide semantics, those are drawn very small and can be safely ignored by considering them a way to connect edges; to support parallelism and to avoid parallelism clutter, we extend it with BPMN parallel gateways (Figure 2c). The complete representation is easily translatable to both BPMN and Petri nets.

Enrichment. The edges and nodes of the map provide an opportunity to enrich the map with information from the event log, such as frequency (FD, CD, PM, FM, IMi-C, ILP-C), performance metrics (FD, CD, PM), data, resources, and deviations (PM, IMi-C, ILP-C) (the latter helping towards evaluation). A perfect process exploration tool would support all of them, and even more, as many measures can provide valuable insight. For now, we demonstrate that these metrics contributing to zoomability can be added, by visualising frequency on the nodes and edges; resources and deviations are visualised using other means.

Zoomabililty. All directly-follows based tools we considered support zoomability by filtering. We discuss a few filtering options here, of which the most common, and basic, are to consider only the most frequent paths (CD, PM, FD), and to consider only the most frequent activities or edges (PM, CD, FD, FM).

Another way to filter is on time, for which two options exist: filtering events on timestamp (FD, CD, PM), which results in a map valid for the chosen interval, and animation (FD, CD, PM, FM), which results in a time-based overlay of the overall map. Animation in these four tools is realised by showing tokens, representing cases, flowing over the edges of the map.

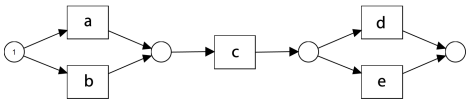


Figure 3: Map discovered by IMi-C from a log in which b and e were not executed in a single trace, which is not shown.

Most tools we considered (PM, CD, FD, FM, IMi-C) have problems discovering and visualising long-distance dependencies, i.e. showing how a choice in the process influences a choice later in the process. For instance, consider the log $L_l = [\langle a, c, e\rangle^{100}, \langle a, c, d\rangle^{100}, \langle b, c, d\rangle^{100}]$, in which b and e are never executed in a single case. This precision information might be interesting, but, as exemplified by Figures 1 and 3, cannot be derived directly from the output of any tool we considered. In most tools (PM, CD,

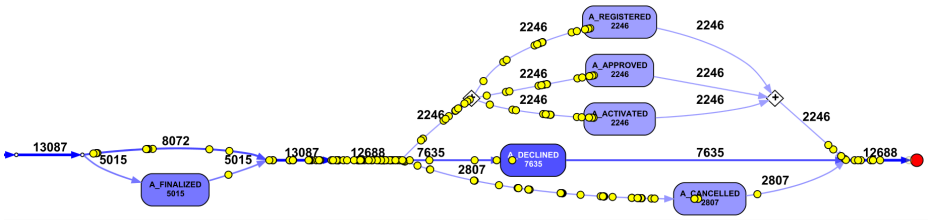


Figure 4: IvM (excerpt). Log animation on part of the BPIC12 log [8].

FD, IMi-C, ILP-C), it is possible to filter all traces not going through b, after which it can be noted that e disappears or is never used. However, some tools make inspecting a *model* difficult by replacing the model with a new one on filtering.

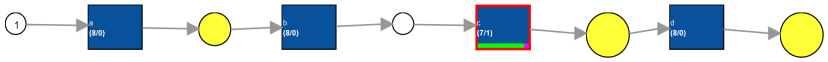
Ideally, a process exploration tool supports as many easily accessible filters as possible. In IvM, we implemented three filters: 1) frequent paths, 2) frequent activities and 3) specific activities. To streamline long-distance dependency inspection, specific activities (3) can be filtered by clicking on nodes in the graph. Moreover, animation was implemented; Figure 4 shows a screenshot.

Evaluation and Deviations. As explained in Section 2, it is important that a process exploration tool enables the evaluation of a model with respect to a log. We analyse evaluation in the existing tools using three levels: model, activity and event. On the model level, there is a single number for an entire model; on the activity level, evaluation is possible on activities or other parts of the model; on the event level, for each event in each trace evaluation is enabled.

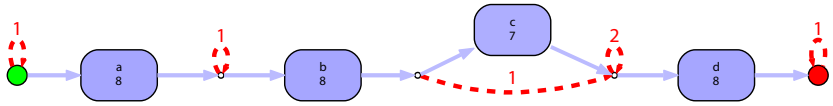
The tools CD, FM and FD provide some model level evaluation by means of their parameters. For instance, FD allows to set a percentage of most frequent paths that should be visualised, giving an estimation of fitness. These measures provide little guidance on the quality of the map. Better model-level fitness metrics are given by FM and PM, indicating what percentage of the events in the log have a corresponding edge in the map. A detailed event-level view is available (FM) that shows for each event in each trace whether it has such a corresponding edge.

The chains IMi-C and ILP-C first compute a model-log alignment, after which the results are visualised in a plug-in common to both. Such an alignment, given a trace t and a model M, is intuitively a best guess of what run of M could have produced t (minimising the number of deviations between the trace and the path through the model). An alignment consists of *synchronous moves*, in which M and t agree on the step taken; *model moves*, in which M took a step and t did not; and *log moves*, in which the t took a step and M did not. For more information about alignments, please refer to [3]. Both IMi-C and ILP-C provide model level statistics as well as activity-level based aggregations (see Figure 5a). An event-level visualisation similar to FM's is available in another plug-in.

Instead of using alignments to visualise deviations, they could be used to repair the model [10]. While repairing a model, the model is updated to allow steps at the position of log moves; model moves are accounted for using circumvention constructs. After repair, perfect fitness is guaranteed, but precision can only deteriorate. Given that process exploration should enable a user to find the right balance between quality dimensions, we cannot use model repair directly.



(a) IMi-C. Non-white places contain log moves and their size indicates the frequency. The third activity has model moves as is indicated by a bar and a thick border.



(b) IvM. Dashed edges denote deviations: self-edges are log moves; the bypassing edge is a model move.

Figure 5: Visualisation of deviations.

Therefore, in IvM we combine the ideas of model repair and alignment visualisation: we perform a model repair, however do not apply it to the model, but add it to the visualisation of the model in dashed/red edges. (we reduce the information about log moves to frequencies for readability reasons) Figure 5b shows an example; if all dashed/red edges would be transformed to normal edges, the model would have perfect fitness, which suits a deviations visualisation.

Ideally, a process exploration tool should enable evaluation on all three levels, thereby providing *zoomable evaluation*. We implemented both the event and the activity-level; the event-level visualisation is similar to the one used in FM.

4 Comparison

In this section, we compare IvM to existing process discovery tools in two ways: 1) we summarise the feature comparison of Section 3 and 2) perform a case study on two real-life examples. Table 1 contains the feature comparison. Most features were introduced in Section 3. [5] [6]

Case Study. In this section, we compare the tools used in this paper on two real-life logs: a log of a financial institution (BPIC12) [8], and a log from a building permit approval process of a Dutch municipality (WABO1BB) [5] [7]. All tools were applied using their default settings.

The BPIC12 log was filtered to only contain the 23 'complete' activities; Figure 6 shows the results of process exploration tools applied to it. These figures exemplify problems of tools we tested: Figures 6c (CD), 6d (PM) and 6e (ILP-C) provide little information by their omnipresence of edges; Figures 6a (FD) and 6b (FM) could be useful for analysis, but conclusions should be drawn carefully: note that in FD, from

[5] 'Local tool' denotes whether the tool can run on the machine of the user; 'Representational bias' refers to the class of models that can be discovered with a tool.

[6] Remarks in Table 1: (1) lower bound on fitness (2) vector screenshot export broken; (3) vector screenshot results in embedded bitmap; (4) PM provides a genetic 'thorough' miner, but that does not guarantee termination; we excluded it from the comparison; (5) available in a separate plug-in; (6) perfect fitness until a filter is applied; (7) could possibly be achieved by writing PQL queries.

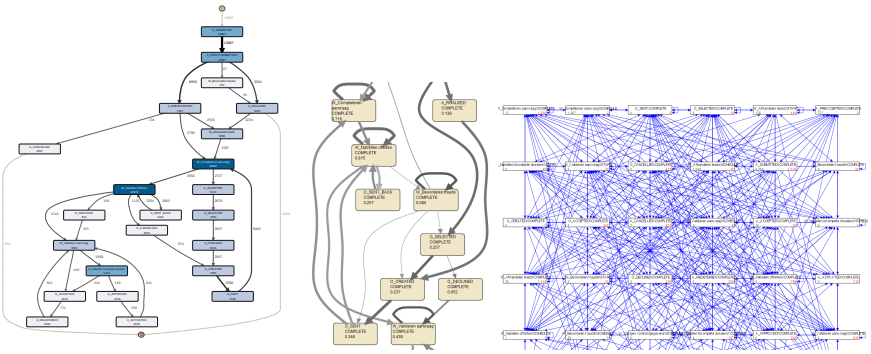
[7] The WABO1BB log has been published between submission and acceptance or this paper.

Table 1: Feature comparison of process discovery tools.

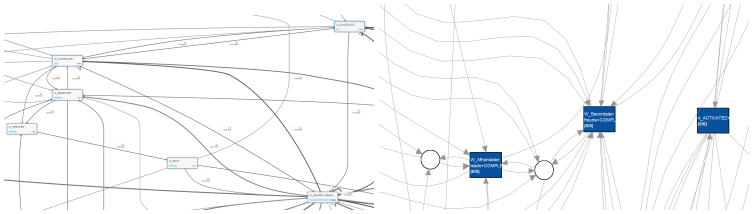
		FD	CD	FM	PM(4)	IvM	IMi-C	ILP-C
	Log import from XES	✓	✗	✓	✗	✓	✓	✓
	Log import from CSV/XLS	✓	✓	✓	✓	✓	✓	✓
	Map export to vector image	✓	✗(2)	✗(3)	✗(2)	✓	✗(3)	✗(3)
	Local tool	✓	✗	✓	✓	✓	✓	✓
Semantics	Executable semantics	✗	✗	✗	✓	✓	✓	✓
	Guaranteed soundness	-	-	-	✓	✓	✓	✗
	Guaranteed perfect fitness (6)	-	-	-	✓	✓	✓	✓
	Best-possible precision	-	-	-	✗	✗	✗	✓
	Representational bias ∋ parallelism	-	-	-	✗	✓	✓	✓
	Representational bias ⩾ ILP-C	-	-	-	✗	✗	✗	✓
	Representational bias ⩾ process trees	-	-	-	✗	✓	✓	✗
	Model export to Process Tree	-	-	-	✗	✓	✓	✗
	Model export to Petri net	-	-	-	✗	✓	✓	✓
	Avoid parallelism-clutter	✓	✗	✓	✗	✓	✓	✓
Zoomability	Frequency enrichment	✓	✓	✓	✓	✓	✓	✓
	Performance enrichment	✓	✓	✗	✓	✗	✓(5)	✓(5)
	Path frequency filter	✓	✗(7)	✗	✓	✓	✓	✗
	Activity/edge frequency filter	✓	✓	✓	✓	✓	✗	✗
	Specific activity/edge filter	✓	✓	✓	✓	✓	✓	✓
	Timestamp filter	✓	✓	✗	✓	✗	✗	✗
	Performance filter	✓	✗(7)	✗	✓	✗	✗	✗
	Animation	✓	✓	✓	✓	✓	✗	✗
Evaluation	Model level	✗	✗	✗	✓(1)	✗	✓	✓
	Activity level	✗	✗	✗	✓	✓	✓	✓
	Event level	✗	✗	✗	✗	✓	✓(5)	✓(5)
	Model repair-semantics	✗	✗	✗	✓	✓	✗	✗
Speed	Immediate parameter feedback	✓	✓	✓	✓	✓	✗	✗
	Long-distance dependency filter without model replacement	✗	✓	✗	✗	✓	✗	✗

only six activities the bottom/end state can be reached, and in both maps it is unclear what the presence or absence of edges actually *means*. As IvM and IMi-C use a similar discovery algorithm and both visualise deviations, their outputs closely resemble one another. However, considering the speed aspect of process exploration, it took 10 screens of parameters/pop-ups to obtain Figure 6g (IMi-C), and *none* to obtain Figure 6f (IvM).

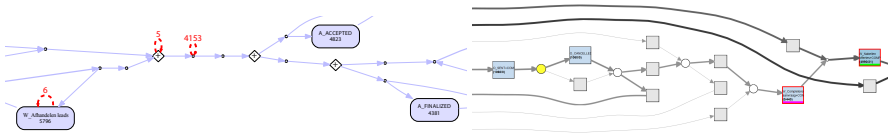
The WABO1BB log was filtered to only contain the 22 'BB' activities. Given the sensitive nature of this log, we were not allowed to upload it to cloud services at time of writing; CD had to be excluded from the analysis. Figure 7 shows the results. Figures 7a (FD) and 7c (FM) are again useful for analysis, but should be read with care: it is clear that the map deviates from the log as activity 01_BB_730 occurred 13 times and has only 4 outgoing edges in FD, but it is not clear how and where the map deviates. Figure 7b (PM) is a quite readable model, however some of the most-used activities appear to be parallel. The model returned by ILP-C (Figure 7d) is not a workflow net and replay requires some log and model moves (this could be avoided using the 'empty after completion' parameter). Figures 7e and 7f show excerpts of the similar models



(a) FD. (b) FM (excerpt). (c) CD (excerpt).



(d) PM (excerpt). (e) ILP-C (excerpt).



(f) IvM (excerpt). (g) IMi-C (excerpt).

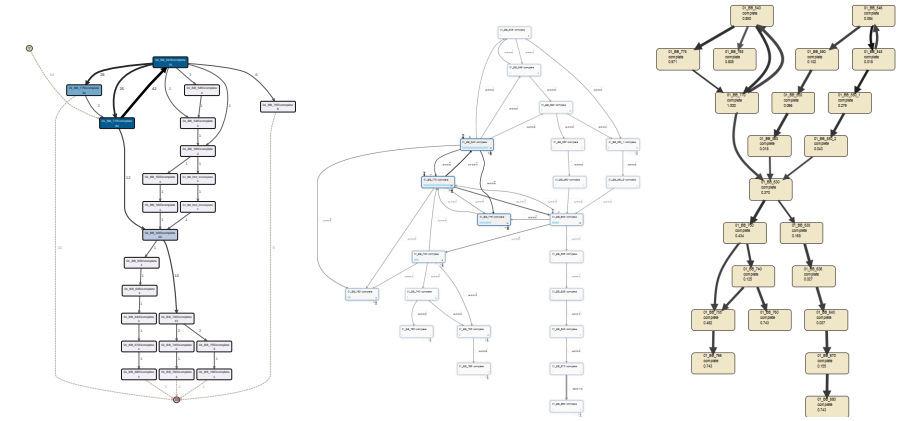
Figure 6: Tools applied to BPIC12, using default settings.

by IMi-C and IvM. The added value of IvM comes when one would like to explore the process and fine-tune the parameters; IMi-C and ILP-C require the user to re-run several plug-ins with each adjustment, IvM does not.

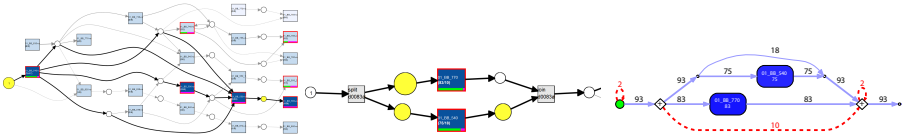
This small case study shows that on the real-life logs we tried, some commercial tools returned difficult-to-interpret maps and others fail to produce readable maps at all. Probably, fine-tuning the parameters might improve the readability of maps, especially in CD and PM. The FM, while academic and a plug-in of ProM, resembles the commercial tools we considered: no executable model semantics but immediate parameter feedback. As IMi-C and IvM use a similar discovery algorithm, the most notable objective differences between IvM and IMi-C are that IvM provides log animation and immediate parameter feedback: *exploring* a process is easy, while IMi-C and ILP-C require a user to leave the visualisation, call a mining plug-in, set all parameters, call an alignment plug-in, and the visualisation plug-in again.

5 Conclusion

In this paper, we identified a gap between commercial and academic process exploration tools. The commercial tools we considered are easy to use and have many features, such



(a) FD. (b) PM. (c) FM.



(d) ILP-C. (e) IMi-C (excerpt). (f) IvM (excerpt).

Figure 7: Tools applied to WABO1BB, using default settings.

as log animation, immediate parameter feedback and extensive filtering options, but the process maps created either do not show parallelism or have no executable semantics and deviations to the maps cannot be computed. Academic tools often create maps with executable semantics, and deviations can be analysed in detail using replay and alignment techniques. However, features important for the exploration of processes are missing and existing tool chains require many steps, thus making exploration tedious and non-interactive.

We introduced a process exploration tool: the Inductive visual Miner (IvM). It aims to bridge this gap between academic and commercial process exploration tools. IvM immediately discovers an initial model, computes deviations and shows these to the user, using a new visualisation that allows for the animation of the traces of a log. IvM is not as feature-rich or scale-oriented as some of the commercial tools, but shows that it is possible to use powerful academic techniques in a user-friendly package. We hope that IvM will inspire commercial vendors to consider models with executable semantics and support deviation analysis.

For future work, one could consider the fast computation of near-optimal alignments. This paper focused on visualising fitness deviations; precision and generalisation problems could be visualised as well, such as in [17]. Furthermore, the Evolutionary Tree Miner [6] could be integrated to obtain an intuitive interactive guided miner.

Acknowledgement. We thank Robin Wolffensperger for his contributions to the positioning of log moves.

References

1. van der Aalst, W.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
2. van der Aalst, W., Weijters, A., Maruster, L.: Workflow mining: Discovering process models from event logs. IEEE Trans. Knowl. Data Eng. 16(9), 1128–1142 (2004)
3. Adriansyah, A.: Aligning Observed and Modeled Behavior. Ph.D. thesis, Eindhoven University of Technology (2014)
4. Bautista, A., Wangikar, L., Kumail Akbar, S.: Process mining-driven optimization of a consumer loan approvals process - The BPIC 2012 challenge case study. In: Business Process Management Workshops. pp. 219–220 (2012)
5. Buijs, J.: Environmental permit application process ('wabo'), CoSeLoG project - municipality 1 (2014), http://dx.doi.org/10.4121/uuid:c45dcbe9-557b-43ca-b6d0-10561e13dcb5
6. Buijs, J., van Dongen, B., van der Aalst, W.: A genetic algorithm for discovering process trees. In: IEEE Congress on Evolutionary Computation. pp. 1–8. IEEE (2012)
7. Buijs, J., van Dongen, B., van der Aalst, W.: On the role of fitness, precision, generalization and simplicity in process discovery. In: On the Move to Meaningful Internet Systems: OTM 2012, pp. 305–322. Springer (2012)
8. van Dongen, B.: BPI Challenge 2012 Dataset (2012), http://dx.doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f
9. van Dongen, B., de Medeiros, A., Verbeek, H., Weijters, A., van der Aalst, W.: The ProM Framework: A new era in process mining tool support. ICATPN 3536, 444–454 (2005)
10. Fahland, D., van der Aalst, W.: Repairing process models to reflect reality. In: BPM'12. LNCS, vol. 7481, pp. 229–245. Springer (2012)
11. Günther, C., van der Aalst, W.: Fuzzy mining–adaptive process simplification based on multi-perspective metrics. Business Process Management pp. 328–343 (2007)
12. Günther, C., Rozinat, A.: Disco: Discover your processes. In: BPM (Demos). CEUR Workshop Proceedings, vol. 940, pp. 40–44. CEUR-WS.org (2012)
13. Leemans, S., Fahland, D., van der Aalst, W.: Discovering block-structured process models from event logs containing infrequent behaviour. In: Business Process Management Workshops. pp. 66–78 (2013)
14. Leemans, S., Fahland, D., van der Aalst, W.: Discovering block-structured process models from incomplete event logs. In: Petri nets 2014. pp. 91–110. Springer (2014)
15. de Leoni, M., van der Aalst, W.: Data-aware process mining: discovering decisions in processes using alignments. In: SAC. pp. 1454–1461. ACM (2013)
16. Ramezani, E., Fahland, D., van der Aalst, W.: Where did I misbehave? Diagnostic information in compliance checking. In: BPM. Lecture Notes in Computer Science, vol. 7481, pp. 262–278. Springer (2012)
17. Rozinat, A.: Process Mining: Conformance and Extension. Ph.D. thesis, Eindhoven University of Technology (2010)
18. Schimm, G.: Process miner - a tool for mining process schemes from event-based data. In: JELIA. LNCS, vol. 2424, pp. 525–528. Springer (2002)
19. Solé, M., Carmona, J.: Process mining from a basis of state regions. In: Petri Nets. LNCS, vol. 6128, pp. 226–245. Springer (2010)
20. Weijters, A., Ribeiro, J.: Flexible Heuristics Miner. In: CIDM. pp. 310–317. IEEE (2011)
21. van der Werf, J., van Dongen, B., Hurkens, C., Serebrenik, A.: Process discovery using integer linear programming. Fundam. Inform. 94(3-4), 387–412 (2009)
22. Wynn, M., Rozinat, A., van der Aalst, W., ter Hofstede, A., Fidge, C.: Process mining and simulation. In: Modern Business Process Automation, pp. 437–457. Springer (2010)