A FRAMEWORK FOR COMPARING PROCESS MINING ALGORITHMS

Philip Weber, Behzad Bordbar, Peter Tiňo

School of Computer Science
University of Birmingham, B15 2TT, UK
P.Weber,B.Bordbar,P.Tino@cs.bham.ac.uk

Basim Majeed

Etisalat BT Innovation Centre (EBTIC)
Khalifa University, Abu Dhabi UAE
Basim.Majeed@bt.com

ABSTRACT

There are many process mining algorithms with different theoretical foundations and aims, raising the question of how to choose the best for a particular situation.

A framework is proposed for objectively comparing algorithms for process discovery against a known ground truth, with an implementation using existing tools.

Results from an experimental evaluation of five algorithms against basic process structures confirm the validity of the approach. In general, numbers of traces for mining are predictable from the structure and probabilities in the model, but there are some algorithm-specific differences.

***Index Terms*—** Business data processing, Modeling, Algorithms

1. INTRODUCTION

Process mining is the extraction of models of business processes, from companies' information systems log files, to enable businesses to better understand and optimise their activities. Models can be graphical, for use by analysts and management, and also formal, for mathematical analysis. This is an active area of research, with many algorithms being created [1] raising the question of how to choose an algorithm for a particular situation. This is a problem both for businesses using process mining, and for researchers evaluating new developments. There is a need for methods for objectively comparing process mining algorithms against known characteristics of business process models and logs, in terms of what can be re-discovered and how much data is required to do so.

In this paper a framework is presented for the evaluation of process mining algorithms against a known ground truth. This includes a method for systematic comparison, and a body of experimental data describing the behaviour of algorithms with typical process structures. An implementation of the method is tested using existing tools. In summary, CPN Tools[1] is used to create and simulate a ground truth in the form of artificial process models consisting of known structures, to produce sets of test process logs. The ProM framework[2] is used to mine process models from these using well-known algorithms, and to check the resulting models for conformance to the ground truth, to find the number of traces required to enable high confidence in being able to re-discover the original model.

The results from tests on a set of basic process structures show that in general the number of traces required can, for at least these simple structures, be predicted from the structure of, and probabilities in, the model. For some structures and algorithms, the results diverge from the expectations suggested by these characteristics. These discrepancies are in line with expectations from the literature, confirming the validity of the method.

Process mining is introduced in section 2, and the problem is described in more detail in section 3. The framework is presented in section 4. Sections 5 and 6 discuss a sample of the experimental testing and results. Some related and future work is outlined in sections 7 and 8.

2. PRELIMINARIES

Various perspectives on a business process can be mined [1]. In this paper the focus is on the control-flow perspective, understanding what business activities take place and in what order. This information can inform activities such as Business Process Management (BPM), troubleshooting, improving efficiency, and so on.

We consider process discovery, rather than compliance, performance or other aspects. An underlying business process model M is assumed, which is the reality of the way the business operates, perhaps different from what is thought. This controls what activities take place. Events relating to activities' occurrence are recorded in logs and can be extracted or reformatted as process traces σ, each of which is a sequence of activities representing one single enactment of the whole process, effectively a sample from the model M. The process mining problem is to 're-discover' a process model M' from a log L of traces and to analyse it for its conformance to the original model M: whether it allows all of, and only, the valid event traces.

3. PROBLEM DESCRIPTION

There are many different approaches to process mining. We refer to the review in [1], and add recent references. Local methods look at local relations between activities in

[1]`http://cpntools.org/`
[2]`http://processmining.org/`

logs (α, α^{++}, Heuristics Miner), while global approaches build and refine a model based on the whole log (Genetic Mining, Region Mining [2], Fuzzy Miner [3]).

Different algorithms have their own specialisms, e.g. α is proven to be able to mine models that adhere to the restrictions of Structured Workflow Nets (SWF-nets) [4], but not mine implicit dependencies or handle noisy logs well; Heuristics Miner uses frequencies and parametrisation to handle noise; while Genetic Process Mining can mine complex and noisy logs, but is resource intensive. More recent approaches focus on managing complex real-world models or noisy logs using clustering and abstraction, e.g. at the trace [5] or activity [6] level.

The problem is the need for a method to objectively compare process mining algorithms, to enable informed choice of which to use in a situation. Which algorithms work 'best', in what circumstances? Which results are the 'best', and can this be quantified in terms of accuracy or confidence [7]?

4. AN EVALUATION FRAMEWORK

A conceptual framework is proposed within which process mining algorithms can be compared objectively. Next, we will describe the outline of the method and a sketch of our implementation.

4.1. Method

The framework is as follows:

1. design a ground truth process model M containing known structures;
2. optionally, estimate the expected number of traces required, to target the experiments;
3. simulate enacting M to produce artificial process logs $\{L_1, L_2, \ldots\}$ with increasing numbers of traces, plus a "very large" log L_*, assumed to be complete (contain all possible process behaviour, distributed approximately according to the underlying reality);
4. mine 're-discovered' models $\{M'_1, M'_2, \ldots\}$ from the logs, using selected algorithms;
5. assess the ability of $\{M'_1, M'_2, \ldots\}$ to 'fit' all the behaviour in L_*, to find the minimum number of traces n for M to be re-discovered with high probability;
6. average over multiple simulations and test data sets.

4.2. Implementation

We implemented and tested the framework using existing tools (Figure1): 1) CPN Tools is used to design Petri Net representations of artificial process models, allowing control over the path probabilities, and 2) with the addition of the ProM CPN Library [8], is used to simulate the models, to produce MXML log file fragments. 3) Log fragments can be merged using the ProMimport CPN Tools plug-in, or with a simple script. 4) A script is used to guide the interactive running of stages 1-3 to produce multiple sets of test logs, independently randomly sampled.

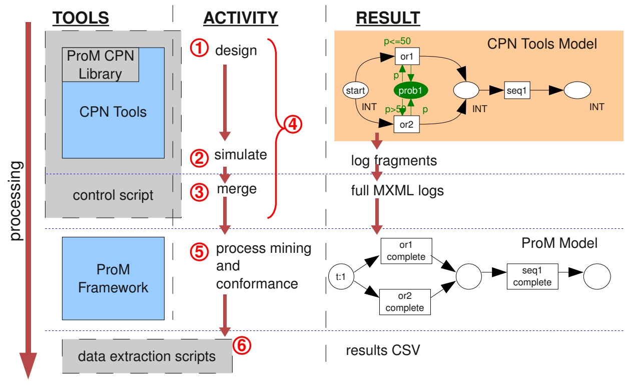


Figure 1. Implementation of Evaluation Framework.

Mining and analysis (5) is a manual process using the ProM framework. For each test, one small log file is mined with one algorithm and the resultant process model (converted, and) exported as a Petri Net Kernel file. The ground truth log file is then loaded into ProM and linked to this Petri Net. Next, the Conformance Checker plug-in [9] is run to measure the mined model against the underlying process described by the large log. 6) The collected results can then be analysed. Data recorded include each algorithm's specific reports, such as the characteristics of the discovered Petri Net, plus the Conformance Checker's assessment of the model's fitness, "behavioural appropriateness" (under-fitting) and "structural appropriateness" (over-fitting and generalisation).

This initial implementation offers much scope for automation to improve speed and accuracy.

5. EXPERIMENTATION

A set of experiments was carried out using the framework described in section 4 using artificial process models comprised of basic structures and sequences and nesting of like structures. Mining was carried out using the α, α^{++}, Heuristics Miner (HM), Genetic Miner (GM) and Region Miner (RM) algorithms. These were chosen because they are fundamental algorithms, commonly used, implemented in the ProM framework, and represent significantly differing approaches to process mining.

A comprehensive set of basic structures and combinations were tested: 2- and 5-way XOR (choice) and AND (parallel) splits/joins, singly and nested to depth 3, and two and three in sequence; single-task loops, executed ≥ 0 and ≥ 1 times; nested single-task loops; and "implicit dependencies" [4, Fig. 7] over 1 or 2 tasks, and nested. While many other structures can be found in 'real-world' business process models, such as non-exclusive splits, complex loops and various types of synchronisation, not all can be straightforwardly represented by Structured Workflow Nets (a Petri Net subset minimal for representing business processes, and mineable by the α algorithm [4]).

Structure	Predict.	Notes
5-way XOR	14	>20 traces required.
3 two-way XOR in seq.	23	α^{++}, RM only 7.
2, 3 two-way AND in seq.	5	HM and GM needed over $15-20$.
Single 5-way AND	120	only 25 needed, except RM which was unable to mine.
Nested AND	5	>50 needed.
Nested loop	71	79 needed except HM failed even with 200 traces.
Nested implicit dep.	11	>100 needed (consistent mining not achieved).

Table 1. Results differing from prediction (HM = Heuristics Miner, RM = Region Miner, GM = Genetic Miner).

5.1. Estimating Numbers of Traces

A simplistic approach to estimating the number of traces is to use the probability of the least probable path through the model, and assert that if that has been seen, probably all other paths will have been seen also. This is not the whole truth, but provides a starting point for experimentation. For example, the number of traces for a 95% probability of discovering a two-way XOR split with path probabilities 0.1 and 0.9, is the number required for a $\leq$ 5% probability of only seeing the 0.9 path: $n \approx \frac{\ln 0.05}{\ln 0.9} < 29$.

5.2. Test Process

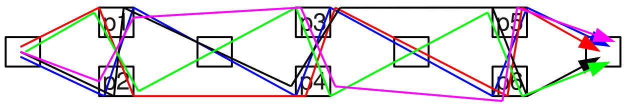
The process described in section 4.2 was followed. In most cases, the number of traces was predicted for 95% confidence in mining a 100% fit model consistently. Due to the manual nature of the test process, only 10 tests were carried out for each model and algorithm, and the conformance results averaged over this set. Rather more samples are needed for greater statistical validity.

For each test model, a 'large' noise-free sample of 5000 random traces was generated by CPN Tools. From these, traces were randomly selected to build the MXML test log files. For each model that was mined, the Fitness, Behavioural and Structural Appropriateness measures, and the number of models which reported 100% Fitness were recorded and compared with predictions.

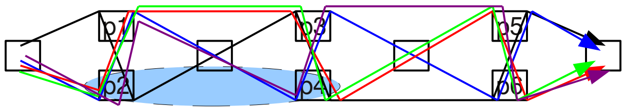
6. RESULTS

Space does not permit full results to be presented, so only results showing differences between the algorithms, or between the predicted and actual number of traces required, are reported (table 1). Further analysis is needed for some of these results, but three main issues arise.

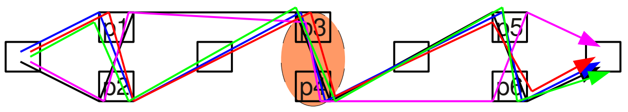
Incorrect Estimation: When estimating the number of traces required for a model, the behaviour of the algorithm and path probabilities in the model need to be taken into account. The low prediction for the 5-way XOR split/join was due to estimation as if the only way for it to fail was for one path of probability 0.2 to be missed, whereas in fact all 5 paths can cause failure in this way. Similarly these algorithms only need to see relations between pairs of tasks in a parallel structure [4], rather than all possible task sequences, so 120 traces estimated for the 5-way AND split is an over-estimate.



(a) All routes present, mineable by all tested algorithms.



(b) Missing path between two pairs, HM and GM not capable of mining.



(c) Pair in one order only (AND), also breaks α and α^{++}.

Figure 2. Traces through AND and XOR sequential constructs and their effect on mineabilty.

Algorithm-Specific Behaviour: Investigating the log files suggests that differences between the algorithms for XOR and AND sequences are explained by how many of the 'transitions' between the paths in one structure to those in the next, they need to see (Figure2). RM was unable to complete the mining, or display the model, for the 5-way AND split. ProM log messages implied that a complex model had been discovered, suggesting that it was unable to infer the parallelism.

HM failed to achieve 100% average fitness for the nested loop model. Further investigation is needed, but this may be due to setting parameters for finding detail in small logs, causing explicit paths to be detected instead of loops, or it may be related to the conversion to Petri Net.

The nesting of AND split/join structures is possibly an overly artificial model design. However, it increases the amount of parallelism considerably and only the Region Miner was able to re-discover the model consistently.

Other Issues: The low estimate for the nested loop is probably due to the way in which the simulated model was designed, which prevented the loops from executing truly independently. HM and α are known not to be able to mine implicit dependencies. The models returned by these algorithms were assessed as 100% fit, but penalised by the behavioural measures for allowing too much extra behaviour. This suggests that improved metrics would still be useful. The nesting of implicit dependencies caused problems for all algorithms, because all tasks in the nested structure are in parallel with tasks in the outer structure, suggesting that many more traces would be needed.

7. RELATED WORK

There is a large body of literature since the mid-1990s for process mining, recently reviewed in [1].

In [7] the need for a comprehensive process mining framework, to include process test data, tools and comparison methods, is discussed. Different evaluation methods are described and tested, and the need highlighted to be able to mine with confidence in the accuracy of the results. The work described in this paper fits with this framework, providing a new evaluation method and proposing a body of empirical results based on process structures.

Conformance analysis and metrics for evaluating process models are further discussed in [9], introducing the Conformance Checker plug-in to ProM. Testing with regard to process structures is described in [10], in the context of testing the Genetic Mining algorithm against certain structures. Some comparison of the behaviour of various algorithms on different models is reported in [11], using the Minimum Description Length principle to encode a log and Petri net, comparing the compression of the log by the model, with the simplicity of the model.

In [12] CPN Tools and the ProM plug-in is used to create and simulate Petri Nets, controlling probabilities, using ProMimport to load the resultant logs to ProM. The inverse process is discussed in [13], along with problems with determining probabilities of paths through the process.

8. CONCLUSIONS

We discussed an outstanding problem in Process Mining, how to choose a mining algorithm, and proposed a framework for objective evaluation; designing, simulating and comparing artificial process models of known structure, against a ground truth. An implementation using existing tools was tested on a set of basic process structures.

Experimentation compared the behaviour of some algorithms with basic process structures, and suggested that the number of traces required for confidence in mining the correct model can in general be predicted from the structure of, and probabilities in, the model. This confirms the validity of the method, with differences being in line with expectations from the literature. Particular results show some algorithms performing differently from predicted, for some structures, suggesting that prior knowledge of the business process should influence the choice of algorithm.

The results indicate that testing based on process structures is worthwhile, as there are differences between the algorithms to be discovered and explained. Many more results need to be collected to build up a comprehensive body of reference data, to enable the comparison of algorithms to inform choice. The framework also needs to be automated and placed on a firm theoretical foundation.

9. REFERENCES

[1] A. Tiwari, C.J. Turner, and B. Majeed, "A review of business process mining: state-of-the-art and future trends," *Bus. Process Manage. J. (UK)*, vol. 14, no. 1, pp. 5 – 22, 2008.

[2] W.M.P. van der Aalst, V. Rubin, B.F. van Dongen, E. Kindler, and C.W. Günther, "Process mining: A two-step approach using transition systems and regions," *Inf. Syst.*, vol. 34, no. 3, pp. 305–327, 2009.

[3] Christian W. Günther and Wil M. R. van der Aalst, "Fuzzy mining – adaptive process simplification based on multi-perspective metrics," *Business Process Management, Proceedings*, vol. LNCS 4714, pp. 328–343, 2007.

[4] W. van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: discovering process models from event logs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 9, pp. 1128–42, 2004.

[5] Minseok Song, Christian W. Gunther, and Wil M. P. Van Der Aalst, "Trace clustering in process mining," in *Lecture Notes in Business Information Processing*, Milano, Italy, 2009, vol. 17 LNBIP, pp. 109 – 120.

[6] Christian W. Gunther, Anne Rozinat, and Wil M. P. Van Der Aalst, "Activity mining by global trace segmentation," in *Lecture Notes in Business Information Processing*, Ulm, Germany, 2010, vol. 43 LNBIP, pp. 128 – 139.

[7] A. Rozinat, A.K. Alves 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," 2007.

[8] A. K. Alves De Medeiros and C. W. Günther, "Process Mining: Using CPN Tools to Create Test Logs for Mining Algorithms," in *Proceedings of the Sixth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*, 2005, pp. 177–190.

[9] A. Rozinat and W.M.P. van der Aalst, "Conformance checking of processes based on monitoring real behavior," *Inf. Syst.*, vol. 33, no. 1, pp. 64–95, 2008.

[10] 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*, vol. 14, no. 2, pp. 245–304, 2007.

[11] T. Calders, C. W. Günther, M. Pechenizkiy, and A. Rozinat, "Using minimum description length for process mining," in *SAC '09: Proceedings of the 2009 ACM symposium on Applied Computing*, New York, NY, USA, 2009, pp. 1451–1455, ACM.

[12] A. Rozinat, R. S. Mans, M. Song, and W. M. P. van der Aalst, "Discovering colored Petri nets from event logs," *Int. J. Softw. Tools Technol. Transf.*, vol. 10, no. 1, pp. 57–74, 2008.

[13] A. Rozinat, R. S. Mans, M. Song, and W. M. P. van der Aalst, "Discovering simulation models," *Inf. Syst.*, vol. 34, no. 3, pp. 305–327, 2009.