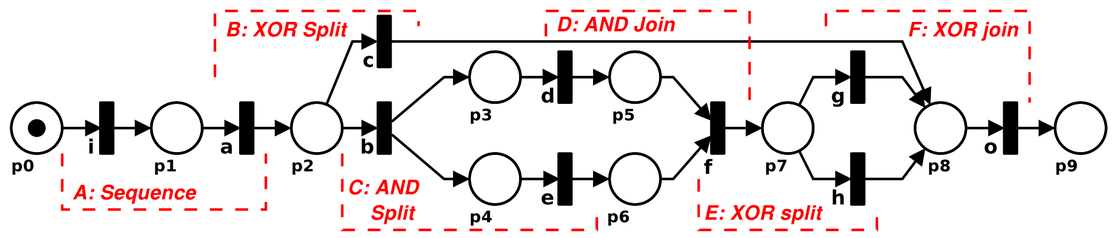
Process Mining in Non-Stationary Environments

Phil Weber, Peter Tiňo and Behzad Bordbar — School of Computer Science, University of Birmingham, UK

EPSRC Engineering and Physical Sciences Research Council

Process Mining

Process mining [1] is the discovery and analysis of models of business processes, from event logs, often represented by Petri nets (PN).



Process mining is used to understand, for example:

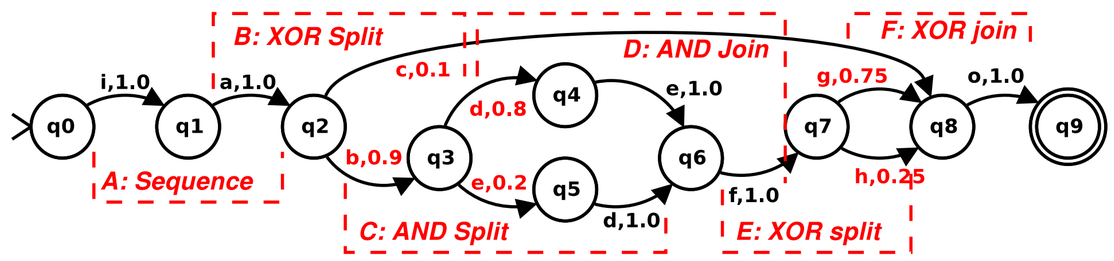
- what activities, resources are involved, and how are they related?
- what affects performance, what decision rules control process flow?
- predict process outcomes and simulate changes.

A Probabilistic Framework

We proposed [2] a probabilistic framework for the **analysis of process mining algorithms**, to rigorously investigate questions such as:

- How much data is needed for mining?
- How accurate are the results?
- How does the algorithm converge?

Activities are modelled by symbols from a finite alphabet Σ, business processes as distributions over sequences $x \in \Sigma^+$, process mining algorithms learn these distributions, succinctly represented by Probabilistic Deterministic Finite Automata (PDFA) [4].



The amount of data an algorithm needs to correctly learn process structures (splits, joins, etc.) — and thus full models — can be predicted from its behaviour and the probabilities in the model.

Example Application to Alpha Mining Algorithm

The Alpha Algorithm [3] uses relations between pairs of activities to construct a Petri net:

$$P_\alpha(a \to_n b) = \left(\mathbf{1} - \pi(\mathbf{ba})\right)^{\mathbf{n}} - \left(\mathbf{1} - \pi(\mathbf{ab}) - \pi(\mathbf{ba})\right)^{\mathbf{n}} \quad \textit{(sequence)},$$

$$P_\alpha(a \parallel_n b) = \mathbf{1} - \left(\mathbf{1} - \pi(\mathbf{ab})\right)^{\mathbf{n}} - \left(\mathbf{1} - \pi(\mathbf{ba})\right)^{\mathbf{n}} + \left(\mathbf{1} - \pi(\mathbf{ab}) - \pi(\mathbf{ba})\right)^{\mathbf{n}} \quad \textit{(parallel)}, \text{ and}$$

$$P_\alpha(a \#_n b) = \left(\mathbf{1} - \pi(\mathbf{ab}) - \pi(\mathbf{ba})\right)^{\mathbf{n}} \quad \textit{(no relation)}.$$

($\pi(ab)$ is the probability of tasks a,b consecutively in a process trace. $P_\alpha(a \to_n b)$ the probability of Alpha 'discovering' a and b in sequence).

We extend to structures, e.g. for an XOR split from task a to multiple options $b_1, b_2, \ldots$ (e.g. structure B in the models above).

$$P_\alpha\left(a \to_n (b_1 \# \ldots \# b_m)\right) \leq \prod_{1 \leq i \leq m} P_\alpha(a \to_n b_i) \times \prod_{1 \leq i < j \leq m} P_\alpha(b_i \#_n b_j).$$

And to full models, e.g. for Petri net PN:

$$P_\alpha(PN) = P_\alpha(i \to_n a) \times P_\alpha\left(a \to_n (b \#_n c) | i \to_n a\right) \times P_\alpha\left(b \to_n (d \parallel_n e) | a \to_n (b \#_n c)\right) \times \ldots.$$

Experimentation confirms convergence as predicted (Figure1) and gives insights into the algorithm's learning behaviour (e.g. Figure2).

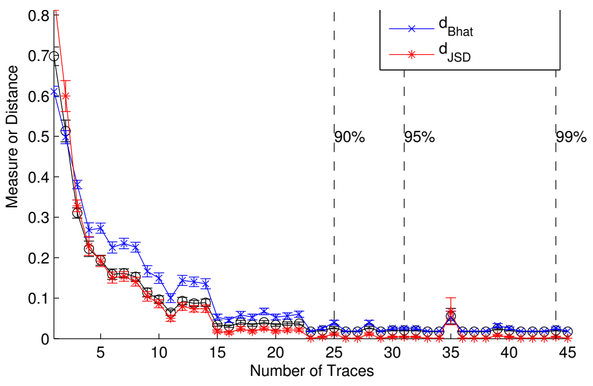


Figure 1: Convergence.

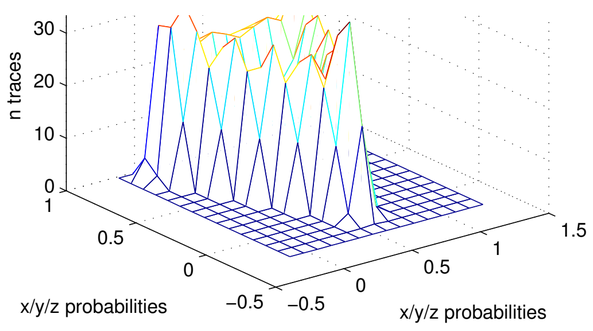


Figure 2: Mining XOR Splits.

Detecting Process Change

We use this analysis and simple tests (e.g. Chi^2) to detect significant change in a running process $\mathcal{M}$ logging its activity (without noise) to log file $\mathcal{W}$. At each detection, we re-estimate number of traces needed.

With confidence (probability $1 - \epsilon$ for desired confidence level $0 < \epsilon \ll 1$) the mined model is correct, so discovery of significant change indicates that the underlying model has truly changed.

1. Obtain $P_{\mathcal{M}'}$, initial estimate of ground truth distribution, from model $\mathcal{M}'$ mined from over-estimate n_0 traces.
2. Analyse $\mathcal{M}'$ to determine number of traces n for confident mining.
3. Mine repeatedly using sliding window of most recent n traces from $\mathcal{W}$, to obtain new distribution $P_{\mathcal{M}''}$.
4. Test difference between $P_{\mathcal{M}''}$ and ground truth estimate $P_{\mathcal{M}'}$.
5. If significant difference, re-estimate ground truth $P_{\mathcal{M}'}$ and n traces.
6. Wait n traces before recommencing change detection.

Framework: Experimental Results

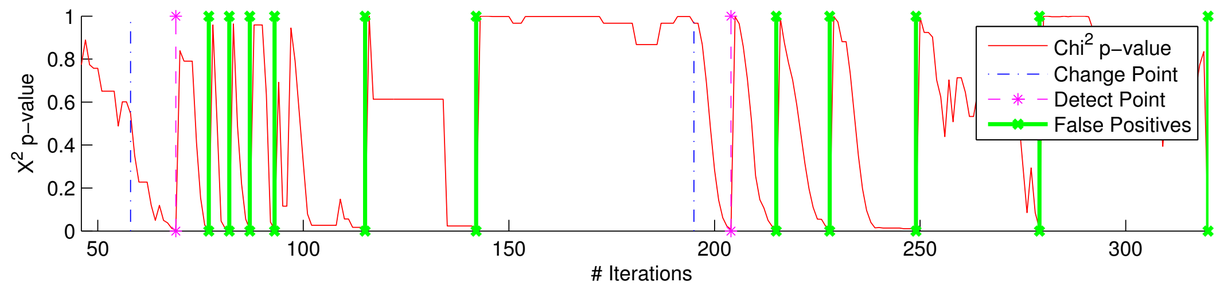
We introduced a number of process changes (see table):

(a) 'large' change in probability of exclusive (XOR) split
(b) 'small' change in probability of parallel split,
(c) change of parallel split to XOR, (d) removal of arc.

	Optimal Sample				Large Sample			
Change	Sample	Detect	KL	p-val	Sample	Detect	KL	p-val
(a)	271	22	0.026	0.007	500	18	0.013	0.034
(b)	45	16	0.163	0.010	500	47	0.013	0.040
(c)	45	15	0.167	0.007	500	195	0.015	0.043
(d)	45	49	0.421	0.004	500	393	0.016	0.034

In the table, Sample traces were used for mining, change was detected in Detect iterations, KL and p-val record the Kullback-Leibler Divergence and Chi^2 p-value between new and previous model estimate. Optimal Sample results used the estimated minimal sample sizes; Large Sample used excessively large samples.

Detection is optimised using our method. With over-large samples, change is detected but not in a timely manner. In a fast-changing environment, if we do not wait before recommencing change detection, false positives occur until the log is fully populated from the new model:



Discussion and Conclusions

To the best of our knowledge, this is the first study to consider process mining in non-stationary environments in an online manner, in a principled way. We estimate (with a given confidence level) the number of traces needed for mining, enabling confidence that discovered change is true rather than an artefact of the log files.

This allows recovery of the set of changed process models in use over time. Also, whereas process mining typically uses non-probabilistic representations such as Petri nets, we are able to discover change that is only apparent in the probabilities in the model.

References

[1] W. M. P. van der Aalst and A. J. M. M. Weijters, *Process mining: a research agenda, Computers and Industry*, vol. 53, no. 3, pp. 231–244, 2004.

[2] Weber, P., Bordbar, B., and Tiňo, P.. *A principled approach to the analysis of process mining algorithms. In Yin, H., Wang, W., and Rayward-Smith, V. J. (eds.), IDEAL 2011*, LNCS vol. 6936, pp. 474–481. Springer, 2011.

[3] 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.

[4] Vidal, E., Thollard, F., de la Higuera, F., Casacuberta, F., and Carrasco, R. C. *Probabilistic Finite-State Machines - Part I. IEEE Trans. Pattern Anal.*, 27(7):1013 – 25, 2005.

{p.weber,b.bordbar,p.tino}@cs.bham.ac.uk