Automated Process Discovery

Sander J. J. Leemans

Definitions

An *event log* contains a historical record of the steps taken in a business process. An event log consists of *traces*, one for each case, customer, order, etc. in the process. A trace contains *events*, which represent the steps (*activities*) that were taken for a particular case, customer, order, etc.

An example of an event log derived from an insurance claim handling process is $[\langle$receive claim, check difficulty, decide claim, notify customer$\rangle^{10}$, $\langle$receive claim, check difficulty, check fraud, decide claim, notify customer$\rangle^{5}]$. This event log consists of 15 traces, corresponding to 15 claims made in the process. In 10 of these traces, the claim was received, its difficulty assessed, the claim was decided and the customer was notified.

A *process model* describes the behaviour that can happen in a process. Typically, it is represented as a Petri net Reisig (1992) or a BPMN model (OMG).

A Petri net consists of places, which denote the states the system can be in, and transitions, which denote the state changes of the system. For instance, Figure 1 shows an example of a Petri net. This net starts with a *token* in place p_1. *Firing* transition a removes the token from p_1 and puts tokens in p_2 and p_3. This denotes the execution of the activity a in the process. Then, transitions b and c can fire independently, each consuming the token of p_2 or p_3 and producing a token in p_4 or p_5. Next, the silent transition t fires and puts a token in p_6. As t is a silent transition, no corresponding activity is executed in the process. Finally, either e or f can be fired, putting a token in p_7 and ending the process.

A *workflow net* is a Petri net with an initial place (without incoming arcs), a final place (without outgoing arcs) and every place and transition lying on a path between these places. The behaviour of

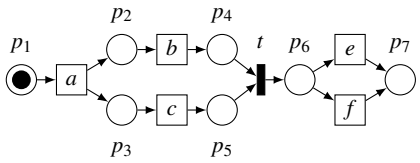

Fig. 1 Example of a Petri net.

a workflow net is clear: a token is put in the initial place, and every sequence of transitions firings that leads to a token in the final place and nowhere else, is a trace of the behaviour of the net.

A workflow net is *sound* if the net is free of deadlocks, unexecutable transitions and other anomalies van der Aalst (2016). A workflow net is *relaxed sound* if there is a sequence of transition firings that lead to a token in the final place and nowhere else.

Automated Process Discovery

Organisations nowadays store considerable amounts of data: in many business processes such as for booking a flight, lodging an insurance claim or hiring a new employee, every step is supported and recorded by an information system. From these information systems, event logs can be extracted, which contain the steps that were taken for a particular customer, booking, claim, etc. Process mining aims to derive information and insights from these event logs.

Many process mining techniques depend on the availability of a process model. Process models can be elicited by hand, however this can be a tedious and error-prone task. Instead, if event logs are available, these can be used to discover a process model automatically.

In this chapter, the research field of algorithms that automatically discover process models from event logs is described. First, quality criteria for models are discussed, and how algorithms might have to tradeoff between them. Second, process discovery algorithms are discussed briefly.

Quality Criteria & Tradeoffs

The quality of a discovered model can be assessed using several concepts: whether it possesses clear semantics, whether it is simple, how well it represents the event log and how well it represents the process.

Semantics & Soundness

As a first quality criterion, the behaviour described by the model should be clear. That is, it should be clear which traces the model can produce. If the returned model is a Petri net or a BPMN model, this model should be free of deadlocks, unexecutable transitions and other anomalies (it should be *sound* van der Aalst (2016)). While unsound nets can be useful for manual analysis, they should be used with care in automated analyses as, for instance, conformance checking techniques might give unreliable answers or simply not work on unsound nets. At a bare minimum, conformance checking techniques such as alignments Adriansyah (2014) require relaxed sound models.

Simplicity

Second, given two models, all other things equal, the simplest model is usually the best of the two (a principle known as Occam's razor). That is, a model should be as understandable as possible, for instance sma

Log Quality

Third, one can consider the quality of a discovered model with respect to the event log from which it was discovered, to assess whether the model represents the available information correctly. Typically, besides simplicity, three quality dimensions are considered: fitness, precision and generalisation. Fitness expresses the part of the event log that is captured in the behaviour of the process model. Precision expresses the part of the behaviour of the model that is seen in the event log. Generalisation expresses what part of future behaviour will be likely present in the model.

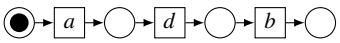

Fig. 2 A process model with low fitness, high precision, low generalisation and high simplicity w.r.t. L.

To illustrate these quality measures, consider the following event log L:

$$[\langle a,d,b\rangle^{50}, \quad \langle a,b,c,d,b\rangle^{20},$$
$$\langle a,b,d,c,b,c,b\rangle^{2}, \quad \langle a,b,c,d,b,c,b\rangle^{2},$$
$$\langle a,b,c,b,d,c,b\rangle, \quad \langle a,b,c,b,c,b,d\rangle,$$
$$\langle a,b,c,b,d,c,b,c,b\rangle, \quad \langle a,b,c\rangle]$$

Figure 2 contains a possible process model for L, which supports only a single trace. This model has a poor fitness, as many traces of L are not part of its behaviour. However, it has a high precision, as the single trace it represents was seen in L. Compared to the event log, this model is not very informative.

An extreme model is shown in Figure 3. This model is a so-called *flower model*, as it allows for all behaviour consisting of a, b, c and d, giving it a low fitness and high precision. Even though this model is simple and certainly generalises, it is completely useless as it does not provide any information besides the presence of a-d in the process.

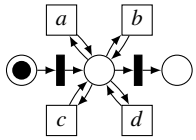

Fig. 3 A process model ("flower model") with high fitness, low precision, high generalisation and high simplicity w.r.t. L.

On the other end of the spectrum is the *trace model*, shown in Figure 4. This model simply lists all traces of L, thereby achieving perfect fitness and precision. However, this model does not generalise the behaviour in the event log, that is, it only shows the traces that were seen in L, and does not provide any extra information.

As a final model, we consider the model shown in Figure 5. This model has a high fitness, precision, generalisation and simplicity. However, the model still does not score *perfect* as the last trace of L, $\langle a,b,c\rangle$, is not captured by this model, which lowers fitness a bit. Furthermore, precision is not perfect as the trace $\langle a,b,c,b,c,d,b\rangle$ is possible in

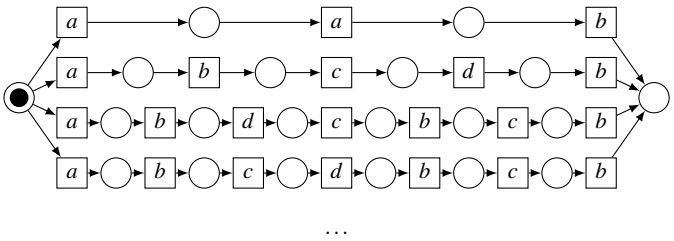

Fig. 4 A process model ("trace model") with high fitness, high precision, low generalisation and low simplicity w.r.t. L.

the model but did not appear in L, which lowers precision.

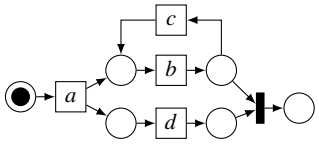

Fig. 5 A process model with high fitness, high precision, high generalisation and high simplicity w.r.t. L.

The models shown for L illustrate that process discovery algorithms might have to tradeoff and strike a balance between quality criteria. For some event logs, a model scoring high on all log-quality measures and simplicity might not exist Buijs et al (2012b). The necessary balance might depend on the use case at hand. For instance, manual analysis where the "main flow" of a process is sought might require the omittance of the last trace of L from the model, yielding a simple and precise model. However, for auditing purposes, one might opt for a perfectly fitting model by including this last trace of L in the behaviour of the model.

Process Quality

A downside of measuring the quality of a model with respect to the event log is that an event log contains only *examples* of behaviour of an (unknown) business process rather than the full behaviour, and that the log might contain traces that do not correspond to the business process (*noisy* traces). Therefore, one can also consider how it compares to the process from which the event log was recorded. In the ideal case, the behaviour of the process is rediscovered by a discovery algorithm. That is, the behaviour (language) of the model is the same as the behaviour of the process.

As the business process is assumed to be unknown, whether an algorithm can find a model that is behaviourally equivalent to the process (*rediscoverability*) is a formal property of the algorithm. Without rediscoverability, an algorithm is *unable* to find a model equivalent to the process, which makes the algorithm rather unsuitable to study this process.

Rediscoverability is typically proven using assumptions on the process and the event log, for instance that it is representible as a model in the formalism of the algorithm (Petri nets, BPMN), and for instance that the event log contains enough information and does not contain

too much noise, as well as assumptions on the process.

Process Discovery Algorithms

In this section, a selection of process discovery algorithms is discussed. For each algorithm, the algorithmic idea is described briefly, as well as some general advantages and disadvantages, and where it can be downloaded.

For benchmarks and a more exhaustive overview, please refer to Augusto et al (2017b) (algorithms after 2012) and Weerdt et al (2012) (algorithms before 2012). Not all algorithms can be benchmarked reliably; the selection here contains all benchmarked algorithms of Augusto et al (2017b).

Several of these algorithms are available in the ProM framework van Dongen et al (2005), which is available for download from `http://www.promtools.org`, or in the Apromore suite Rosa et al (2011), which can be accessed via `http://apromore.org`.

The algorithms are discussed in three stages: firstly, algorithms that do not support concurrency, secondly algorithms that guarantee soundness and thirdly the remaining algorithms.

Directly Follows-Based Techniques

As a first set, techniques based on the directly follows relations are discussed. The section starts with an explanation of directly follows graphs, after which some tools that use this concept are listed and the limitations of such techniques are discussed.

In a directly follows graph, the nodes represent the activities of the event log, and the edges represent that an activity is directly followed by another activity in the event log. Numbers on the edges indicate how often this happened. Additionally, a start and an end node denote the events with which traces in the event log start or end. For instance, Figure 6 shows the directly follows graph for our event log L.

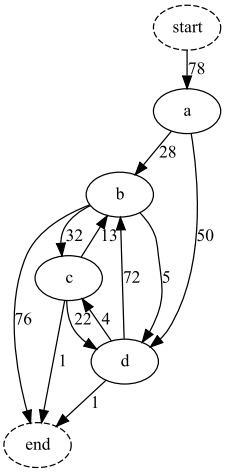

Fig. 6 Directly follows graph of event log L.

For more complicated processes, a directly follows graph might get uncomprehensibly complicated. Therefore, discovery techniques typically filter the directly follows graph, for instance by removing little-occurring edges. Commercial techniques that filter and show directly follows graphs include Fluxicon Disco Fluxicon (2017), Celonis Process Mining Celonis (2017) and ProcessGold Enterprise Platform ProcessGold (2017). Another strategy to reduce complexity, applied by the

Fuzzy Miner Günther and van der Aalst (2007), is to cluster similar activities into groups, thereby providing capabilities to zoom in on details of the process (by clustering less), or to abstract to the main flow of the process by clustering more.

While these graphs are intuitive, it can be challenging to distinguish repetitive and concurrent behaviour, as both manifest as edges forth- and back between activities. For instance, in Figure 6, it seems that b, c and d can be executed repeatedly, while in the log L this never happened for d. In contrast, it *also* seems that b, c and d are concurrent, while in L, b and c are always executed repeatedly. Due to this, directly follows graphs tend to have a low precision and high generalisation: in our example, almost any sequence of b, c and d is included.

Nevertheless, directly follows-based techniques are often used to get a first idea of the process behind an event log.

Soundness-Guaranteeing Algorithms

Soundness is a prerequisite for further automated or machine-assisted analysis of business process models. In this section, soundness or relaxed soundness guaranteeing algorithms are discussed.

Evolutionary Tree Miner

To address the issue of soundness, the Evolutionary Tree Miner (ETM) Buijs et al (2012a) discovers process trees. A process tree is an abstract hierarchical view of a workflow net and is inherently sound.

ETM first constructs an initial population of models; randomly or from other sources. Second, some models are selected based on fitness, precision, generalisation and simplicity with respect to the event log. Third, the selected models are smart-randomly mutated. This process of selection and mutation is repeated until a satisfactory model is found, or until time runs out.

ETM is flexible as both the selection and the stopping criteria can be adjusted to the use case at hand; one can prioritise (combinations of) quality criteria. However, due to the repeated evaluation of models, on large event logs of complex processes, stopping criteria might force a user to make the decision between speed and quality.

Inductive Miner Family

The Inductive Miner (IM) family of process discovery algorithms, like the Evolutionary Tree Miner, discovers process trees to guarantee that all models that are discovered are sound. The IM algorithms apply a recursive strategy: first, the "most important" behaviour of the event log is identified (such as sequence, exclusive choice, concurrency, loop, etc.). Second, the event log is split in several parts, and these steps are repeated until a base case is encountered (such as a log consisting of a single activity). If no "most important" behaviour can be identified, then the algorithms try to continue the recursion by generalising the behaviour in the log, in the worst case ultimately ending in a flower model.

Besides a basic IM Leemans et al (2013a), algorithms exist that focus on filtering noise Leemans et al (2013b), handling incomplete behaviour (when the event log misses crucial information of the process) Leemans et al (2014a), handling lifecycle information of events (if the log contains information of e.g. when activities started and ended) Leemans et al (2015), discovering challenging constructs such as inclusive choice and silent steps Leemans (2017), and handling very large logs and complex processes Leemans et al (2016), all available in the ProM framework.

Several IM-algorithms guarantee to return a model that perfectly fits the event log, and all algorithms are capable of rediscovering the process, assuming that the process can be described as a process tree (with some other restrictions, such as no duplicated activities) and assuming that the event log contains "enough" information. However, due to the focus on fitness, precision tends to be lower on event logs of highly unstructured processes.

All Inductive Miner algorithms are available as plug-ins of the ProM framework, and some as plug-ins of the Apromore framework. Furthermore, the plug-in Inductive visual Miner Leemans et al (2014b) provides an interactive way to apply these algorithms and perform conformance checking.

An algorithm that uses a similar recursive strategy, but lets constructs compete with one another is the Constructs Competition Miner Redlich et al (2014), however its implementation has not been published.

Structured Miner

The Structured Miner (STM) Augusto et al (2016) applies a different strategy to obtain highly block-structured models and to tradeoff the log quality criteria. Instead of discovering block-structured models directly, SM first discovers BPMN models and, second, structures these models. The models can be obtained from any other discovery technique, for instance Heuristics Miner or Fodina, as these models need not be sound. These models are translated to BPMN, after which they are made block-structured by shifting BPMN-gateways in or out, thereby duplicating activities.

STM benefits from the flexibility of the used other discovery technique to strike a flexible balance between log-quality criteria and can guarantee to return sound models. However, this guarantee comes at the price of equivalence (the model is changed, not just restructured), simplicity (activities are duplicated) and speed (the restructuring is $O(n^n)$).

STM is available as both a ProM and an Apromore plugin.

(Hybrid) Integer Linear Programming Miner

The Integer Linear Programming Miner (ILP) van der Werf et al (2009) constructs a Petri net, starting with all activities as transitions and no places, such that every activity can be arbitrarily executed. Second, it adds places using an optimisation technique: a place is only added if it does not remove any trace of the event log from the behaviour of

the model. Under this condition, the behaviour is restricted as much as possible.

ILP focusses on fitness and precision: it guarantees to return a model that fits the event log, and the most precise model within its representational bias (Petri nets, no duplicated activities). However, the ILP miner does not guarantee soundness, does not handle noise and tends to return complex models Leemans (2017).

The first of these two have been addressed in the HybridILPMiner van Zelst et al (2017), which performs internal noise filtering. Furthermore, it adjusts the optimisation step to guarantee that the final marking is always reachable, and, in some cases, returns workflow nets, thereby achieving *relaxed soundness*.

Declarative Techniques

Petri nets and BPMN models express what *can* happen when executing the model. In contrast, declarative models, such as Declare models, express what *cannot* happen when executing the model, thereby providing greater flexibility in modelling. Declare miners such as Maggi et al (2011); Di Ciccio et al (2016); Ferilli et al (2016) discover the constraints of which Declare models consist using several acceptance criteria, in order to be able to balance precision and fitness. However, using such models in practice tends to be challenging Augusto et al (2017b).

Other Algorithms

Unsound models are unsuitable for futher automated processing, however might be useful for manual analysis. In the remainder of this section, several algorithms are discussed that do not guarantee soundness.

α-Algorithms

The first process discovery algorithm described was the α-algorithm van der Aalst et al (2004). The α algorithm considers the directly follows graph and identifies three types of relations between sets of activities from the graph: sequence, concurrency and mutual exclusivity. From these relations, a Petri net is constructed by searching for certain maximal patterns.

The α algorithm is provably Badouel (2012) able to rediscover some processes, assuming that the log contains enough information and with restrictions on the process. In later versions, several restrictions have been addressed, such as: a) no short loops (activities can follow one another directly; addressed in α^+ de Medeiros et al (2004)), b) no long-distance dependencies (choices later in the process depend on choices made earlier; addressed in Wen et al (2006)), c) no non-free-choice constructs (transitions that share input places have the same input places; addressed in α^{++} Wen et al (2007a)), and d) no silent transitions (addressed in $\alpha^{\#}$ Wen et al (2010, 2007b) and in $\alpha^{\$}$ Guo et al (2015)). Furthermore, a variant has been proposed, called the Tsinghua-α Wen et al (2009), that deals with non-atomic event logs. That

is, event logs in which executions of activities take time.

However, these algorithms guarantee neither soundness nor perfect fitness nor perfect precision, the algorithms cannot handle noise and cannot handle incompleteness. Furthermore, the α algorithms might be less fast on complex event logs, as typically they are exponential. Therefore, the α-algorithms are not very suitable to be applied to real-life logs.

Little Thumb Weijters and van der Aalst (2003) extends the α algorithms with noise-handling capabilities: instead of considering binary activity relations, these relations are derived probabilistically and then filtered according to a user-set threshold.

Causal-Net Miners

The Flexible Heuristics Miner (FHM) Weijters and Ribeiro (2011) uses the probabilistic activity relations of Little Thumb and focuses on soundness. To solve the issue of soundness, FHM returns causal nets, a model formalism in which it is defined that non-sound parts of the model are not part of the behaviour of the net.

The Fodina algorithm vanden Broucke and Weerdt (2017) extends FHM with long-distance dependency support and, in some cases, duplicate activities. The Proximity miner Yahya et al (2016) extends FHM by incorporating domain knowledge. For more algorithms using causal nets, please refer to Weerdt et al (2012); Augusto et al (2017b).

Even though causal nets are sound by definition, they place the burden of soundness checking on the interpreter/user of the net, and this still does not guarantee, for instance, that every activity in the model can be executed. Therefore, translating a causal net to a Petri net or BPMN model for further processing does not guarantee soundness of the translated model.

FHM, Fodina (`http://www.processmining.be/fodina`) and Proximity Miner (`https://sourceforge.net/projects/proxi-miner/`) are all available as ProM plug-ins and/or Apromore plug-ins.

Split Miner

To strike a different balance in log-quality criteria compared to IM that favours fitness, while improving in speed over ETM, Split Miner (SPM) Augusto et al (2017a) preprocesses the directly follows graph before constructing a BPMN model. In the preprocessing of directly follows graphs, first, loops and concurrency are identified and filtered out. Second, the graph is filtered in an optimisation step: each node must be on a path from start to end, the total number of edges is minimised, while the sum of edge frequencies is maximised. Then, splits and joins (BPMN gateways) are inserted to construct a BPMN model.

SPM aims to improve over the precision of IM and the speed of ETM for real-life event logs. The balance between precision and fitness can be adjusted in the directly follows-optimisation step, which allows users to adjust the amount of noise filtering. However, the returned models are not guaranteed to be sound (proper completion is not guaranteed), and several OR-joins might be inserted, which increases complexity.

SPM is available as a plug-in of Apromore and as a stand-alone tool via https://doi.org/10.6084/m9.figshare.5379190.v1.

Conclusion

Many process mining techniques require a process model as a prerequisite. From an event log, process discovery algorithms aim to discover a process model, this model preferably having clear semantics, being sound, striking a user-adjustable balance between fitness, precision, generalisation and simplicity, and having confidence that the model represents the business process from which the event log was recorded. Three types of process discovery algorithms were discussed: directly follows-based techniques, soundness-guaranteeing algorithms and other algorithms, all targetting a subset of these quality criteria.

In explorative process mining projects, choosing a discovery algorithm and its parameters is a matter of repeatedly trying *soundness-guaranteeing* algorithms, evaluating their results using conformance checking and adjusting algorithm, parameters and event log as new questions pop up van Eck et al (2015).

Cross-References

- Event log cleaning for business process analytics
- Conformance checking
- Process model repair
- Decision discovery in business processes

References

(2011) Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2011, part of the IEEE Symposium Series on Computational Intelligence 2011, April 11-15, 2011, Paris, France, IEEE, URL http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=5937059

van der Aalst W, Weijters A, Maruster L (2004) Workflow mining: Discovering process models from event logs. IEEE Trans Knowl Data Eng 16(9):1128–1142

van der Aalst WMP (2016) Process Mining - Data Science in Action, Second Edition. Springer

Adriansyah A (2014) Aligning Observed and Modeled Behavior. PhD thesis, Eindhoven University of Technology

Augusto A, Conforti R, Dumas M, Rosa ML, Bruno G (2016) Automated discovery of structured process models: Discover structured vs. discover and structure. In: Comyn-Wattiau I, Tanaka K, Song I, Yamamoto S, Saeki M (eds) Conceptual Modeling - 35th International Conference, ER 2016, Gifu, Japan, November 14-17, 2016, Proceedings, Lecture Notes in Computer Science, vol 9974, pp 313–329, DOI 10.1007/978-3-319-46397-1_25, URL http://dx.doi.org/10.1007/978-3-319-46397-1_25

Augusto A, Conforti R, Dumas M, Rosa ML (2017a) Split miner: Discovering accurate and simple business process models from event logs. In: IEEE International Conference on Data Mining, New Orleans, LA, URL https://eprints.qut.edu.au/110153/

Augusto A, Conforti R, Dumas M, Rosa ML, Maggi FM, Marrella A, Mecella M, Soo A (2017b) Automated discovery of process models from event logs: Review and benchmark. CoRR abs/1705.02288, URL http://arxiv.org/abs/1705.02288, 1705.02288

Badouel E (2012) On the α-reconstructibility of workflow nets. In: Haddad S, Pomello L (eds) Application and Theory of Petri Nets - 33rd International Conference, PETRI NETS 2012, Hamburg, Germany, June 25-29, 2012. Proceedings, Springer, Lecture Notes in Computer Science, vol 7347, pp 128–147, DOI 10.1007/978-3-642-31131-4_8, URL http://dx.doi.org/10.1007/978-3-642-31131-4_8

vanden Broucke SKLM, Weerdt JD (2017) Fodina: A robust and flexible heuristic process discovery technique. Decision Support Systems 100:109–118, DOI 10.1016/j.dss.2017.04.005, URL https://doi.org/10.1016/j.dss.2017.04.005

Buijs JCAM, van Dongen BF, van der Aalst WMP (2012a) A genetic algorithm for discovering process trees. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2012, Brisbane, Australia, June 10-15, 2012, IEEE, pp 1–8, DOI 10.1109/CEC.2012.6256458, URL http://dx.doi.org/10.1109/CEC.2012.6256458

Buijs JCAM, van Dongen BF, van der Aalst WMP (2012b) On the role of fitness, precision, generalization and simplicity in process discovery. In: Meersman R, Panetto H, Dillon TS, Rinderle-Ma S, Dadam P, Zhou X, Pearson S, Ferscha A, Bergamaschi S, Cruz IF (eds) On the Move to Meaningful Internet Systems: OTM 2012, Confederated International Conferences: CoopIS, DOA-SVI, and ODBASE 2012, Rome, Italy, September 10-14, 2012. Proceedings, Part I, Springer, Lecture Notes in Computer Science, vol 7565, pp 305–322, DOI 10.1007/978-3-642-33606-5_19, URL http://dx.doi.org/10.1007/978-3-642-33606-5_19

Celonis (2017) Process Mining. https://www.celonis.com/, [Online; accessed 11-November-2017]

Di Ciccio C, Maggi FM, Mendling J (2016) Efficient discovery of target-branched declare constraints. Inf Syst 56:258–283, DOI 10.1016/j.is.2015.06.009, URL http://dx.doi.org/10.1016/j.is.2015.06.009

van Dongen BF, de Medeiros AKA, Verbeek HMW, Weijters AJMM, van der Aalst WMP (2005) The prom framework: A new era in process mining tool support. In: Ciardo G, Darondeau P (eds) Applications and Theory of Petri Nets 2005, 26th International Conference, ICATPN 2005, Miami, USA, June 20-25, 2005, Proceedings, Springer, Lecture Notes in Computer Science, vol 3536, pp 444–454, DOI 10.1007/11494744_25, URL http://dx.doi.org/10.1007/11494744_25

van Eck ML, Lu X, Leemans SJJ, van der Aalst WMP (2015) PMˆ2 : A process mining project methodology. In: Zdravkovic J, Kirikova M, Johannesson P (eds) Advanced Information Systems Engineering - 27th International Conference, CAiSE 2015, Stockholm, Sweden, June 8-12, 2015, Proceedings, Springer, Lecture Notes in Computer Science, vol 9097, pp 297–313, DOI 10.1007/978-3-319-19069-3_19, URL http://dx.doi.org/10.1007/978-3-319-19069-3_19

Ferilli S, Esposito F, Redavid D, Angelastro S (2016) Predicting process behavior in woman. In: Adorni G, Cagnoni S, Gori M, Maratea M (eds) AI*IA 2016: Advances in Artificial Intelligence - XVth International Conference of the Italian Association for Artificial Intelligence, Genova, Italy, November 29 - December 1, 2016, Proceedings, Springer, Lecture Notes in Computer Science, vol 10037, pp 308–320, DOI 10.1007/978-3-319-49130-1_23, URL https://doi.org/10.1007/978-3-319-49130-1_23

Fluxicon (2017) Disco. http://fluxicon.com, [Online; accessed 11-November-2017]

Günther C, van der Aalst W (2007) Fuzzy mining–adaptive process simplification based on multi-perspective metrics. Business Process Management pp 328–343

Guo Q, Wen L, Wang J, Yan Z, Yu PS (2015) Mining invisible tasks in non-free-choice constructs. In: Motahari-Nezhad HR, Recker J, Weidlich M (eds) Business Process Management - 13th International Conference, BPM 2015, Innsbruck, Austria, August 31 - September 3, 2015, Proceedings, Springer, Lecture Notes in Computer Science, vol 9253, pp 109–125, DOI 10.1007/978-3-319-23063-4_7, URL http://dx.doi.org/10.1007/978-3-319-23063-4_7

Leemans S (2017) Robust process mining with guarantees. PhD thesis, Technische Universiteit Eindhoven

Leemans SJJ, Fahland D, van der Aalst WMP (2013a) Discovering block-structured process models from event logs - a constructive approach. In: Colom JM, Desel J (eds) Application and Theory of Petri Nets and Concurrency - 34th International Conference, PETRI NETS 2013, Milan, Italy, June 24-28, 2013. Proceedings, Springer, Lecture Notes in Computer Science, vol 7927, pp 311–329, DOI 10.1007/978-3-642-38697-8_17, URL http://dx.doi.org/10.1007/978-3-642-38697-8_17

Leemans SJJ, Fahland D, van der Aalst WMP (2013b) Discovering block-structured process models from event logs containing infrequent behaviour. In: Lohmann N, Song M, Wohed P (eds) Business Process Management Workshops - BPM 2013 International Workshops, Beijing, China, August 26, 2013, Revised Papers, Springer, Lecture Notes in Business Information Processing, vol 171, pp 66–78, DOI 10.1007/978-3-319-06257-0_6, URL http://dx.doi.org/10.1007/978-3-319-06257-0_6

Leemans SJJ, Fahland D, van der Aalst WMP (2014a) Discovering block-structured process models from incomplete event logs. In: Ciardo G, Kindler E (eds) Application and Theory of Petri Nets and Concurrency - 35th International Conference, PETRI NETS 2014, Tunis, Tunisia, June 23-27, 2014. Proceedings, Springer, Lecture Notes in Computer Science, vol 8489, pp 91–110, DOI 10.1007/978-3-319-07734-5_6, URL http://dx.doi.org/10.1007/978-3-319-07734-5_6

Leemans SJJ, Fahland D, van der Aalst WMP (2014b) Process and deviation exploration with Inductive visual Miner. In: Limonad L, Weber B (eds) Proceedings of the BPM Demo Sessions 2014 Co-located with the 12th International Conference on Business Process Management (BPM 2014), Eindhoven, The Netherlands, September 10, 2014., CEUR-WS.org, CEUR Workshop Proceedings, vol 1295, p 46, URL http://ceur-ws.org/Vol-1295/paper19.pdf

Leemans SJJ, Fahland D, van der Aalst WMP (2015) Using life cycle information in process discovery. In: Reichert M, Reijers HA (eds) Business Process Management Workshops - BPM 2015, 13th International Workshops, Innsbruck, Austria, August 31 - September 3, 2015, Revised Papers, Springer, Lecture Notes in Business Information Processing, vol 256, pp 204–217, DOI 10.1007/978-3-319-42887-1_17, URL http://dx.doi.org/10.1007/978-3-319-42887-1_17

Leemans SJJ, Fahland D, van der Aalst WMP (2016) Scalable process discovery and conformance checking. Software & Systems Modeling special issue:1–33, DOI 10.1007/s10270-016-0545-x, URL http://dx.doi.org/10.1007/s10270-016-0545-x

Maggi FM, Mooij AJ, van der Aalst WMP (2011) User-guided discovery of declarative process models. In: DBL (2011), pp 192–199, DOI 10.1109/CIDM.2011.5949297, URL http://dx.doi.org/10.1109/CIDM.2011.5949297

de Medeiros AKA, van Dongen BF, van der Aalst WMP, Weijters AJMM (2004) Process mining for ubiquitous mobile systems: An overview and a concrete algorithm. In: Baresi L, Dustdar S, Gall HC, Matera M (eds) Ubiquitous Mobile Information and Collaboration Systems, Second CAiSE Workshop, UMICS 2004, Riga, Latvia, June 7-8, 2004, Revised Selected Papers, Springer, Lecture Notes in Computer Science, vol 3272, pp 151–165, DOI 10.1007/978-3-540-30188-2_12, URL http://dx.doi.org/10.1007/978-3-540-30188-2_12

(OMG) OMG (2011) Business process model and notation (BPMN) version 2.0. Tech. rep., Object Management Group (OMG)

ProcessGold (2017) Enterprise Platform. http://processgold.com/en/, [Online; accessed 11-November-2017]

Redlich D, Molka T, Gilani W, Blair GS, Rashid A (2014) Constructs competition miner: Process control-flow discovery of bp-domain constructs. In: Sadiq SW, Soffer P, Völzer H (eds) Business Process Management - 12th International Conference, BPM 2014, Haifa, Israel, September 7-11, 2014. Proceedings, Springer, Lecture Notes in Computer Science, vol 8659, pp 134–150,

DOI 10.1007/978-3-319-10172-9_9, URL http://dx.doi.org/10.1007/978-3-319-10172-9_9

Reisig W (1992) A primer in Petri net design. Springer Compass International, Springer

Rosa ML, Reijers HA, van der Aalst WMP, Dijkman RM, Mendling J, Dumas M, García-Bañuelos L (2011) APROMORE: an advanced process model repository. Expert Syst Appl 38(6):7029–7040, DOI 10.1016/j.eswa.2010.12.012, URL https://doi.org/10.1016/j.eswa.2010.12.012

Weerdt JD, Backer MD, Vanthienen J, Baesens B (2012) A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs. Inf Syst 37(7):654–676, DOI 10.1016/j.is.2012.02.004, URL http://dx.doi.org/10.1016/j.is.2012.02.004

Weijters AJMM, van der Aalst WMP (2003) Rediscovering workflow models from event-based data using little thumb. Integrated Computer-Aided Engineering 10(2):151–162, URL http://content.iospress.com/articles/integrated-computer-aided-engineering/ica00143

Weijters AJMM, Ribeiro JTS (2011) Flexible heuristics miner (FHM). In: DBL (2011), pp 310–317, DOI 10.1109/CIDM.2011.5949453, URL http://dx.doi.org/10.1109/CIDM.2011.5949453

Wen L, Wang J, Sun J (2006) Detecting implicit dependencies between tasks from event logs. In: Zhou X, Li J, Shen HT, Kitsuregawa M, Zhang Y (eds) Frontiers of WWW Research and Development - APWeb 2006, 8th Asia-Pacific Web Conference, Harbin, China, January 16-18, 2006, Proceedings, Springer, Lecture Notes in Computer Science, vol 3841, pp 591–603, DOI 10.1007/11610113_52, URL http://dx.doi.org/10.1007/11610113_52

Wen L, van der Aalst WMP, Wang J, Sun J (2007a) Mining process models with non-free-choice constructs. Data Min Knowl Discov 15(2):145–180, DOI 10.1007/s10618-007-0065-y, URL http://dx.doi.org/10.1007/s10618-007-0065-y

Wen L, Wang J, Sun J (2007b) Mining invisible tasks from event logs. In: Dong G, Lin X, Wang W, Yang Y, Yu JX (eds) Advances in Data and Web Management, Joint 9th Asia-Pacific Web Conference, APWeb 2007, and 8th International Conference, on Web-Age Information Management, WAIM 2007, Huang Shan, China, June 16-18, 2007, Proceedings, Springer, Lecture Notes in Computer Science, vol 4505, pp 358–365, DOI 10.1007/978-3-540-72524-4_38, URL http://dx.doi.org/10.1007/978-3-540-72524-4_38

Wen L, Wang J, van der Aalst WMP, Huang B, Sun J (2009) A novel approach for process mining based on event types. J Intell Inf Syst 32(2):163–190, DOI 10.1007/s10844-007-0052-1, URL http://dx.doi.org/10.1007/s10844-007-0052-1

Wen L, Wang J, van der Aalst WMP, Huang B, Sun J (2010) Mining process models with prime invisible tasks. Data Knowl Eng 69(10):999–1021, DOI 10.1016/j.datak.2010.06.001, URL http://dx.doi.org/10.1016/j.datak.2010.06.001

van der Werf JMEM, van Dongen BF, Hurkens CAJ, Serebrenik A (2009) Process discovery using integer linear programming. Fundam Inform 94(3-4):387–412, DOI 10.3233/FI-2009-136, URL http://dx.doi.org/10.3233/FI-2009-136

Yahya BN, Song M, Bae H, Sul S, Wu J (2016) Domain-driven actionable process model discovery. Computers & Industrial Engineering 99:382–400, DOI 10.1016/j.cie.2016.05.010, URL https://doi.org/10.1016/j.cie.2016.05.010

van Zelst SJ, van Dongen BF, van der Aalst WMP, Verbeek HMW (2017) Discovering relaxed sound workflow nets using integer linear programming. CoRR abs/1703.06733, URL http://arxiv.org/abs/1703.06733, 1703.06733