View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Online Research Database In Technology

Technical University of Denmark

Streaming Process Discovery and Conformance Checking

Burattin, Andrea

Published in:
Encyclopedia of Big Data Technologies

Link to article, DOI:
10.1007/978-3-319-63962-8_103-1

Publication date:
2018

Document Version
Peer reviewed version

Link back to DTU Orbit

Citation (APA):
Burattin, A. (2018). Streaming Process Discovery and Conformance Checking. In S. Sakr , & A. Zomaya (Eds.), Encyclopedia of Big Data Technologies Springer. DOI: 10.1007/978-3-319-63962-8_103-1

DTU Library
Technical Information Center of Denmark

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Streaming process discovery and conformance checking

Andrea Burattin

Synonyms

Online process mining; online process discovery; online conformance checking.

Definitions

Streaming process discovery, streaming conformance checking, and streaming process mining in general (also known as *online process mining*) are disciplines which analyze event streams to extract a process model or to assess their conformance with respect to a given reference model. The main characteristic of this family of techniques is to analyze events immediately as they are generated (instead of storing them in a log for late processing). This allows to drastically reduce the latency among phases of the BPM lifecycle (cf. Dumas et al (2013)), thus allowing faster process adaptations and better executions.

Overview

A possible characterization of process mining algorithms is based on how they consume event data. Specifically, most of the algorithms focus on a (static) *event log*, however there are algorithms which focus on *event streams*. An event log is a finite sampling of activities observed in a given time frame. An event stream, on the

Andrea Burattin
DTU Compute, Software Engineering, Technical University of Denmark, 2800 Kgs. Lyngby, Denmark. E-mail: andbur@dtu.dk

other hand, is an unbounded *sequence* of events, which contains events as they are executed.

Event streams, in fact, are specific types of data streams, where each data point refers to an event. General data streams have been investigated since many years, mainly to tackle problems such as frequency counting, classification, clustering, approximation, time series analysis and change diagnosis (also known as novelty detection or concept drift detection) as summarized in Widmer and Kubat (1996); Gama (2010); Gaber et al (2005); Aggarwal (2007). To tackle these problems, it is possible to devise techniques in a data- or task-based orientation. *Data-based techniques* aims at extracting a significantly representative finite subset of the data stream, which is used for the analysis. *Task-based techniques*, on the other hand, adapt the computation to this new data modality, in order to minimize time and space complexity of the analysis.

The streams these algorithms have to deal with can be characterized based on their *operations model*. In particular, we might have:

- insert-only stream model (once an element is seen, it cannot be changed);
- insert-delete stream model (cf.n elements can be deleted or updated);
- additive stream model where seen item refers to numerical variables which are just incremented.

In the context of process mining, all available techniques assume the insert-only model: observations refer to activities which have been executed. The data mining literature, for example in Golab and Özsu (2003); Bifet et al (2010), informally, defines data streams as a *fast sequence of data items*. However, in Bifet and Kirkby (2009), several assumptions on the data stream are reported, such as: the data is assumed to have a small and fixed number of attributes; the number of data points is very large; stream concepts (in the process mining context, the *concept* is the model underlying the events being generated) are assumed to be stationary or evolving. These assumptions make the processing of data streams very different from conventional relational models. Specifically, as detailed in (Gama, 2010, Table 2.1), the data elements arrive online, with no control by the system, in an unbounded amount. This imposes the analysis to be incremental: once an element is received it is discarded or analyzed. If it is analyzed it cannot be explicitly retrieved again afterwards (i.e., its information is aggregated and summarized). Additionally, to cope with concept drifts, old observations should be replaced by new ones and it is not possible to have one-time queries but a continuous "querying mechanism" is necessary.

Key Research Findings

This section provides some general ideas on how to tackle the problem of process mining from an event stream. Three general strategies are sketched and, in the upcoming subsections, details regarding actual instantiations of the ideas are reported. Figure 1 depicts a taxonomy of the different approaches investigated so far. In this

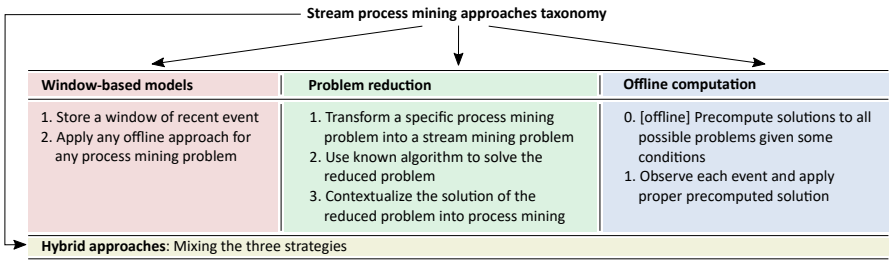

Fig. 1 Taxonomy of the different approaches to solve the different stream process mining problems. For each technique, corresponding general steps are sketched.

Algorithm 1: Window model

Input: S: event stream
M: memory model
max_M: maximum memory
A: additional information (e.g., a reference model), can be $\emptyset$

```
1  forever do
       // Observe a new event
2      e ← observe(S)
       // Memory update
3      if max(M) ≥ max_M then
4          dequeue(M) // Forgetting
5      end
6      insert(M, e)
       // Mining update
7      if perform mining then
           // Memory into event log
8          L ← convert(M)
9          ProcessMining(L, A)
10     end
11 end
```

text we will not focus on hybrid approaches: they are characterized by very heterogeneous solutions and therefore there is no proper generalization possible.

Window models. In order to perform process mining on a stream of event, the simplest approach is to devised a data-based technique to store only the set of most recent events observed and periodically analyze them. Whenever there is no more memory available, the oldest event is discarded. This approach, called *window model* is described in Alg. 1. The algorithm enters a endless loop where, at each iteration, a new event is observed. Then, the system checks whether it is necessary to remove old events or not, and then the new event is inserted. Periodically, the system converts the memory into a standard event log, and classical process mining algorithms are applied on it. The literature, in Babcock et al (2002), identifies at least two memory models capable of storing event: *sequence based* or *timestamp based*. The first approach (which is typically implemented with *sliding windows*) consists

Algorithm 2: Lossy Counting

Input: S: data stream
ε: maximal approximation error

```
1  T ← ∅ // Initially empty set
2  N ← 1 // Number of observed events
3  w ← ⌈1/ε⌉ // Bucket width
4  forever do
5      e ← observe(S)
6      b_curr ← ⌈N/w⌉
       /* Is there a tuple in T with e as first component?          */
7      if e is already in T then
8          Increment the frequency of e in T
9      else
10         Insert (e, 1, b_curr − 1) in T
11     end
12     if N mod w = 0 then
13         forall (a, f, Δ) ∈ T s.t. f + Δ ≤ b_curr do
14             Remove (a, f, Δ) from T
15         end
16     end
17     N ← N + 1
18 end
```

of a FIFO queue of fixed size. The observed events are stored in the window and, once the maximum capacity is reached, the oldest event is removed. In a timestamp based window model the approach is very similar but the memory size is not fixed. Instead, the removal is based on the "age" of the observation: the memory keeps only the events observed within the given time span.

This approach comes with several advantages, such as the possibility to reuse every mining algorithm already available for event logs. However, the memory management is extremely problematic and has a huge impact on the performance of the approach. Specifically, all window-based approaches have poor summarizing capabilities since, for example, duplicate events require two "memory slots", even though they may not provide new information.

Problem reduction. The second possible way of tackling the streaming process mining problems is to employ a task-based technique. For example, it is possible to *reduce* the process mining problem to another well-establish problem and therefore reuse algorithms specifically devised and optimized for the necessity at hand. Of course, it is also possible to devise a completely new algorithm specifically tailored to solve the given situation. This approach, for example, has been used to reduce the problem of streaming process discovery to the *frequency counting problem*. This problem consists of counting the frequencies of given variables over a stream. In order to reduce the process discovery to such problem it is important to understand *what is* a variable in the process mining context, and whether it is possible to identify it.

An example of efficient algorithm for the approximated frequency counting problem is Lossy Counting, here summarized in Alg. 2, and described in Manku and Motwani (2002). The idea is to conceptually split the observed stream in buckets, each with a fixed size. The approach takes as input the stream and the maximal ap-

proximation error on the counting $\varepsilon \in [0,1]$, which drives the size of each bucket. The approach stores entries in a data structure T where each component (e, f, Δ) provides the element e of the stream, the estimated frequency for it f, and the maximum number of times it could have occurred Δ. When a new event is observed in the stream, the algorithm checks if it is already in T and, in case, it increments its counter f by one. Otherwise a new entry in T is created. Periodically (i.e., every time a new conceptual bucket starts) the algorithm cleans the memory, by removing elements not frequently observed. This algorithm has no memory bound. Specifically, the size of the data structure T depends on the stream and on the approximation error. However, a variation of the algorithm to enforce fixed amount of memory is described in Da San Martino et al (2012).

The most relevant benefit of the problem reduction approach is that, once the process mining problem has been reduced to the new one, it is possible to use algorithms already devised for the reduced problem. However, such reduction might not be trivial and all assumptions of the used approach have to be met.

Offline computation. The last approach we present consists of moving the computation of the solutions to the given problem from online setting to offline. In other words, the idea is to identify and solve all sub-problems the stream may provide. Adopting this approach will help us in dealing with all situations we are going to observe, still keeping the complexity of the online processing constant.

The advantage of this approach is the possibility of having extremely expensive solutions "cached" in advance which are then just reused whenever needed. Though, there are several drawbacks: it is not possible to apply this approach to all online process mining problems. Additionally, by computing everything in advance, we lose the possibility of adapting the pre-computed solutions to the contextual information, which might be uniquely specific to the running process instance.

In the upcoming subsections, one example of each technique will be presented and detailed by presenting two process mining activities.

Process Discovery

The first problem we present is the online process discovery. A graphical conceptualization of the problem is reported in Fig. 2. The idea is to have a source which is generating an event stream. This event stream is consumed by a miner which generates a representation of the underlying process model and keeps it up-to-date with respect to the observed behavior.

The first approach available to tackle this problem is reported in Burattin et al (2012, 2014b). In their works, authors employ a problem reduction strategy. Specifically, they reduce the Heuristics Miner algorithm, described in van der Aalst and Weijters (2003), to the Frequency Counting problem. To do that, they assume *direct following relationships* as variables and, by counting them, they are able to use the Heuristics Miner's metric to reconstruct the high level business patterns in terms of Heuristics Net or Petri Net. Authors test different algorithms to solve the frequency

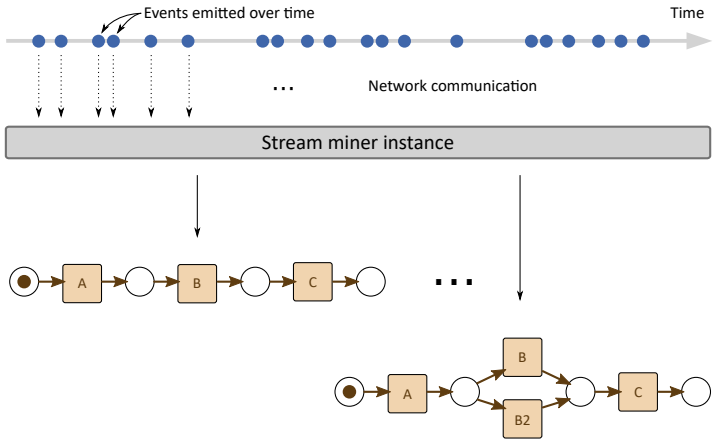

Fig. 2 Conceptualization of the streaming process discovery idea as in Burattin et al (2014b).

counting problem, such as Lossy Counting and Lossy Counting with Budget. As baseline approach, authors use a sliding window mechanism where Heuristics Miner is iteratively applied. The sliding window approach is constantly outperformed by other approaches which provide a better usage of the available resources. A similar approach, called StrProM, tracks the direct following relationships. This approach, presented in Hassani et al (2015), keeps an updated prefix tree by deriving a solution based on Lossy Counting with Budget. The direct following relationships are then used to construct a process model using the set of rules of Heuristics Miner.

As for the previously mentioned approaches, in Maggi et al (2013); Burattin et al (2014a, 2015), authors investigate the problem of discovering a process represented in Declare (cf. Pesic et al (2007)). Specifically, their idea is to instantiate several "replayers", one for each Declare template to mine. Then, specific behavior for each template is implemented. To keep track of the replay statuses, authors use Lossy Counting-based strategies. Authors apply sliding window as baseline and, again, the performances of the Lossy Counting strategies are better with respect to the simple application of offline approaches over a sliding window.

In Redlich et al (2014b) authors present the adaptation of CCM to cope with event streams. The basic idea of CCM (cf. Redlich et al (2014a)) is to identify subsequences of events in order to identify footprints of specific process patterns. In CCM, relevant patters and corresponding footprints are described. Ageing factors are employed to the collected information in order to give more importance to recent behavior.

The most recent work tackling the online process discovery is reported in van Zelst et al (2017b). In their paper, authors generalize previously instantiated concepts: they present an architecture, namely S-BAR, which keeps an updated abstract representation of the stream (e.g., direct follow relationships) and they use it as starting point to infer an actual process model. In their work, authors present the adaptations of different mining algorithms: α (cf. van der Aalst et al (2004)), Heuris-

tics Miner (cf. van der Aalst and Weijters (2003)) and Inductive Miner (cf. Leemans et al (2013)). To show the generability of their abstract representation, authors also show a miner based on Region Theory (cf. van der Aalst et al (2008)). In order to keep their abstraction updated, authors reduce their problem to frequency counting, thus using Lossy Counting, Space Saving (cf. Metwally et al (2005)), and Frequent (cf. Karp et al (2003)).

Conformance Checking

The problem of online conformance checking has received attention in the declarative domain. In this case, it used to be called *operational support* and its aim is to understand whether a set of constraints is being violated or not. To achieve that, in Maggi et al (2011, 2012), authors devise an approach to represent the behavior as an automaton and executions are replayed on it. Additionally, each process instance is labeled with one of 4 possible fulfillment states: permanently/temporarily violated/fulfilled.

Concerning imperative models, online conformance checking received less attention. At present time, only two approaches have been specifically designed to tackle the online conformance checking problem. One approach, described in van Zelst et al (2017a), aims at reusing the concept of *alignment* in order to compute the optimal alignment just for the prefix of the trace seen up to a given point in time. To this end, authors first devise a technique to compute prefix-alignments. Authors prove the optimality of the prefix-alignment they discover. Up to this point, their approach is incremental but not really online (i.e., it is able to work on partial cumulative information but in theory they need infinite memory to back-track in order to find the optimal alignment). To solve this issue, authors mention the possibility to implement memory management, either falling into a window-based model or as problem reduction. Authors, however, do not implement or test either memory management approach. However, they test their technique against different number of backtracking steps required to find the prefix-alignment.

Another conformance checking approach belongs to the offline computation category and is described in Burattin and Carmona (2017); Burattin (2017). In this case, given a Petri Net as input, authors describe a technique to elicit a transition system containing all possible transitions from one marking to another one, even those which are not described by the original model. Each transition in this elicited model is then associated with a cost. Transitions allowed by the original Petri Net have cost 0, all others have cost > 0. This way, by replaying a trace on such transition system and summing the costs, authors show that conformant traces will have cost 0 and non-conformant trace always have cost larger than 0. Additionally, authors ensures the determinism of such transition system and the possibility, from each state, to have one transition for each possible activity. These two last properties guarantees the suitability of the approach for online settings.

Another conformance checking approach is presented in Weber et al (2015). In this case, authors propose a RESTful service which performs a token replay on a BPMN model. In particular, authors implemented a *token pull mechanism*. Authors refer this approach as online primarily because of its interface, while no explicit guarantee is mentioned in terms on memory usage and computational complexity.

Other Applications

Online process mining has been applied also to discover cooperative structures out of event streams. For example, in van Zelst et al (2016), authors are able to process an event stream and update the set of relationships of a cooperative resource network.

Additionally, in order to conduct research on stream process mining, it is useful to simulate an event stream for which we know the actual original source. To achieve that, mainly two approaches are available. The first is a tool called PLG2[1], described in Burattin (2016). It allows to generate a random model or to load a BPMN file and stream events referring to actual executions. The stream is sent over a TCP/IP connection, and the tool allows different configurations in terms of noise and concept drift. The second tool is described in van Zelst et al (2015). This tool allows researchers to design a model – as Petri Net – in CPN[2] and then import it into ProM[3] where the `XESEventStream Publisher` plugin can be used to simulate a stream. Please note that, in this case, the stream exists just within ProM.

Key Applications

Due to the novelty of the topic, we are not aware of any deployment of streaming process mining techniques in real settings. However, more generally, online process mining techniques are relevant in all cases where it is important to have results in real-time, to immediately enact proper responses. In this section, examples related to IT setting will be provided, but business oriented applications are absolutely possible and relevant as well.

Online process discovery might be useful in settings where it is important to analyze immediately the behavior of the system. For example, reconstructing the behavior of the services used in a website might be useful in order to see what is currently under stress and what is going to be used afterwards. Exploiting this information could improve the resource allocation (e.g., upscaling or downscaling the server capacity on the fly).

---

[1] `http://plg.processmining.it/`

[2] `http://www.cpntools.org/`

[3] `http://www.promtools.org/`

Online conformance checking is also useful whenever it is important to immediately detect deviations from reference behavior to enact proper countermeasures. For example, the kernel of an operating system exposes some services for applications. These services should be combined in some specific ways (e.g., a file should be `open()`, then either `write()` or `read()` or both appear, and eventually the file should be `close()`) which represent the reference behavior. If an application is observed strongly violating such behavior it might be an indication of strange activities going on, for example in order to bypass some imposed limitations or privileges.

Future Directions for Research

As previously mentioned, online process mining is a relatively new area of investigation. Some techniques are available for the discovery of the process (both as imperative and declarative model). Very recently, online conformance checking also received attention, but several improvements are still needed.

First of all, techniques should improve their efficiency, for example in terms of memory consumption. This represent an important research direction since, in the online setting, the amount of available memory is fixed, thus representing a key resource. To tackle this problem, it is necessary to work on the summarization capabilities used by the algorithms in order to find more compact ways of storing the same information.

Related to the previous point is the quality and quantity of information and contextual data algorithms are able to extract out of the same event stream. For example, a process discovery algorithm might be extended to extract more complex control flow patters or the algorithm might be modified to return not just the result but the confidence on the provided outcomes.

Additionally, there are more technical issues that algorithms should be able to cope with. For example, dealing with a stream where the arrival time of events do not coincide with their actual execution. In this case, it would be necessary to reorder the list of events belonging to the same process instance before mining them. Please note that, assuming different order implies a sort of different element model of the stream (i.e., it becomes an “insert-delete stream model”, where the order of events can change). Another relevant issue might be the inference of the termination of a process instance.

Cross-References

- Automated process discovery
- Conformance checking
- Declarative process mining

- Definition of data streams
- Introduction to stream processing algorithms

References

van der Aalst WM, Weijters TAJMM (2003) Rediscovering Workflow Models from Event-based Data Using Little Thumb. Integrated Computer-Aided Engineering 10(2):151–162

van der Aalst WM, Weijters TAJMM, Maruster L (2004) Workflow Mining: Discovering Process Models from Event Logs. IEEE Transactions on Knowledge and Data Engineering 16:2004

van der Aalst WM, Günther CW, Rubin V, Verbeek EHMW, Kindler E, van Dongen B (2008) Process mining: a two-step approach to balance between underfitting and overfitting. Software & Systems Modeling 9(1):87–111, DOI 10.1007/s10270-008-0106-z

Aggarwal CC (2007) Data Streams: Models and Algorithms. Advances in Database Systems, Springer US, Boston, MA, DOI 10.1007/978-0-387-47534-9

Babcock B, Babu S, Datar M, Motwani R, Widom J (2002) Models and Issues in Data Stream Systems. In: Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp 1–16, DOI 10.1145/543614.543615

Bifet A, Kirkby R (2009) Data Stream Mining: A Practical Approach. Tech. rep., Centre for Open Software Innovation - The University of Waikato

Bifet A, Holmes G, Kirkby R, Pfahringer B (2010) MOA: Massive Online Analysis Learning Examples. Journal of Machine Learning Research 11:1601–1604

Burattin A (2016) PLG2 : Multiperspective Process Randomization with Online and Offline Simulations. In: Online Proceedings of the BPM Demo Track 2016, CEUR-WS.org

Burattin A (2017) Online conformance checking for petri nets and event streams. In: CEUR Workshop Proceedings, vol 1920

Burattin A, Carmona J (2017) A Framework for Online Conformance Checking. In: Proceedings of the 13th International Workshop on Business Process Intelligence (BPI 2017)., Springer, p (in press)

Burattin A, Sperduti A, van der Aalst WM (2012) Heuristics Miners for Streaming Event Data. ArXiv CoRR URL `http://arxiv.org/abs/1212.6383`

Burattin A, Maggi FM, Cimitile M (2014a) Lights, Camera, Action! Business Process Movies for Online Process Discovery. In: Proceedings of the 3rd International Workshop on Theory and Applications of Process Visualization (TAProViz 2014)

Burattin A, Sperduti A, van der Aalst WM (2014b) Control-flow Discovery from Event Streams. In: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, pp 2420–2427, DOI 10.1109/CEC.2014.6900341

Burattin A, Cimitile M, Maggi FM, Sperduti A (2015) Online Discovery of Declarative Process Models from Event Streams. IEEE Transactions on Services Computing 8(6):833–846, DOI 10.1109/TSC.2015.2459703

Da San Martino G, Navarin N, Sperduti A (2012) A Lossy Counting Based Approach for Learning on Streams of Graphs on a Budget. In: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, AAAI Press, pp 1294–1301

Dumas M, La Rosa M, Mendling J, Reijers HA (2013) Fundamentals of Business Process Management. Springer

Gaber MM, Zaslavsky A, Krishnaswamy S (2005) Mining Data Streams: a Review. ACM Sigmod Record 34(2):18–26, DOI 10.1.1.80.798

Gama J (2010) Knowledge Discovery from Data Streams. Chapman and Hall/CRC, DOI 10.1201/EBK1439826119

Golab L, Özsu MT (2003) Issues in Data Stream Management. ACM SIGMOD Record 32(2):5–14, DOI 10.1145/776985.776986

Hassani M, Siccha S, Richter F, Seidl T (2015) Efficient Process Discovery From Event Streams Using Sequential Pattern Mining. In: 2015 IEEE Symposium Series on Computational Intelligence, pp 1366–1373, DOI 10.1109/SSCI.2015.195

Karp RM, Shenker S, Papadimitriou CH (2003) A simple algorithm for finding frequent elements in streams and bags. ACM Transactions on Database Systems 28(1):51–55, DOI 10.1145/762471.762473

Leemans SJJ, Fahland D, van der Aalst WM (2013) Discovering Block-Structured Process Models from Event Logs - A Constructive Approach. In: Proceedings of Petri Nets, Springer Berlin Heidelberg, pp 311–329, DOI 10.1007/978-3-642-38697-8{_}17

Maggi FM, Montali M, Westergaard M, van der Aalst WM (2011) Monitoring Business Constraints with Linear Temporal Logic : An Approach Based on Colored Automata. In: Proceedings of the 9th international conference on Business process management, Springer Berlin Heidelberg, pp 132–147, DOI 10.1007/978-3-642-23059-2{_}13

Maggi FM, Montali M, van der Aalst WM (2012) An operational decision support framework for monitoring business constraints. In: Proceedingss of 15th International Conference on Fundamental Approaches to Software Engineering (FASE), pp 146–162, DOI 10.1007/978-3-642-28872-2{_}11

Maggi FM, Bose RPJC, van der Aalst WM (2013) A Knowledge-Based Integrated Approach for Discovering and Repairing Declare Maps. In: 25th International Conference, CAiSE 2013, Valencia, Spain, June 17-21, 2013., Springer Berlin Heidelberg, pp 433–448, DOI 10.1007/978-3-642-38709-8{_}28

Manku GS, Motwani R (2002) Approximate Frequency Counts over Data Streams. In: Proceedings of International Conference on Very Large Data Bases, Morgan Kaufmann, Hong Kong, China, pp 346–357

Metwally A, Agrawal D, Abbadi AE (2005) Efficient Computation of Frequent and Top-k Elements in Data Streams. In: Database Theory - ICDT 2005, Springer Berlin Heidelberg, pp 398–412, DOI 10.1007/978-3-540-30570-5{_}27

Pesic M, Schonenberg H, van der Aalst WM (2007) DECLARE: Full Support for Loosely-Structured Processes. In: Proceedings of EDOC, IEEE, pp 287–298, DOI 10.1109/EDOC.2007.14

Redlich D, Molka T, Gilani W, Blair G, Rashid A (2014a) Constructs competition miner: Process control-flow discovery of BP-domain constructs. In: Proceedings of BPM 2014, pp 134–150, DOI 10.1007/978-3-319-10172-9{_}9

Redlich D, Molka T, Gilani W, Blair G, Rashid A (2014b) Scalable dynamic business process discovery with the constructs competition miner. In: Proceedings of the 4th International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2014), vol 1293, pp 91–107

Weber I, Rogge-Solti A, Li C, Mendling J (2015) CCaaS: Online conformance checking as a service. In: Proceedings of the BPM Demo Session 2015, vol 1418, pp 45–49

Widmer G, Kubat M (1996) Learning in the Presence of Concept Drift and Hidden Contexts. Machine Learning 23(1):69–101, DOI 10.1007/BF00116900

van Zelst SJ, van Dongen B, van der Aalst WM (2015) Know What you stream: Generating event streams from CPN models in ProM 6. In: CEUR Workshop Proceedings, pp 85–89

van Zelst SJ, van Dongen B, van der Aalst WM (2016) Online Discovery of Cooperative Structures in Business Processes. In: Proceedings of the OTM 2016 Conferences, Springer International Publishing, pp 210–228

van Zelst SJ, Bolt A, Hassani M, van Dongen B, van der Aalst WM (2017a) Online conformance checking: relating event streams to process models using prefix-alignments. International Journal of Data Science and Analytics DOI 10.1007/s41060-017-0078-6

van Zelst SJ, van Dongen B, van der Aalst WM (2017b) Event stream-based process discovery using abstract representations. Knowledge and Information Systems pp 1–29, DOI 10.1007/s10115-017-1060-2