Ebi - a Stochastic Process Mining Framework

Sander J.J. Leemans[1,*], Tian Li[1,2] and Jan Niklas van Detten[1]

[1]RWTH Aachen, Germany
[2]University of Melbourne, Australia

Abstract

Ebi is an open-source framework for stochastic process mining. Ebi currently contains over 30 techniques related to stochastic process mining, from completeness estimation to analysis, conformance checking, discovery, visualisation and statistical tests, and can be called from the command line.

For technique developers, Ebi allows abstractions on inputs through interfaces ("traits"), such that implemented techniques automatically support a variety of input and output types. Furthermore, Ebi almost completely uses exact arithmetic to avoid precision issues with low values.

Keywords

Stochastic process mining, stochastic process models, event logs

Metadata description	Value
Tool name	Ebi
Current version	1.0
Legal code license	Apache 2.0
Languages, tools and services used	Rust, GraphViz, Rust4pm
Supported operating environment	Windows, Linux, (Mac OS X with self-compilation)
Download/Demo URL	https://www.ebitools.org/
Documentation URL	https://www.ebitools.org/, click `manual`
Source code repository	https://github.com/BPM-Research-Group/Ebi
Screencast video	https://youtu.be/IEeTH3DCZ_0

1. Introduction

Using process mining techniques, business analysts can find inefficiencies in business processes, with the ultimate aim of improving them. For efficient use of process improvement resources, it is important to recognise whether certain to-be-improved behaviour is frequent or rare. Such frequency information is referred to as the *stochastic perspective* in process mining, and techniques that take this stochastic information into account explicitly are called *stochastic process mining* techniques.

Stochastic process mining techniques are scattered across several process mining frameworks, such as Pm4py [1], BupaR [2] and ProM [3], each with its strengths and weaknesses, ranging from computational efficiency to ease of applicability in high-performance computing settings.

ICPM 2024 Tool Demonstration Track, October 14-18, 2024, Kongens Lyngby, Denmark
*Corresponding author.
s.leemans@bpm.rwth-aachen.de (S.J.J. Leemans)
0000-0002-5201-7125 (S.J.J. Leemans)

None of these existing frameworks (i) allows a straightforward application of a particular technique to an input file, (ii) none is easily accessible from the command line, (iii) all require installation and external runtime environments and, most importantly, (iv) none support exact computations throughout.

Ebi is a command line utility focused on stochastic process mining techniques. For end users, it allows direct application of stochastic process mining techniques to files, and writes the output to a file. No need to import, click through options and export, or even to write scripts. In particular, Ebi supports exact computations, which are a necessity in stochastic process mining (see Section 2).

For developers, Ebi is a framework that allows abstractions on inputs through interfaces ("traits"), such that implemented techniques automatically support a variety of input and output types. Furthermore, Ebi provides the means to perform exact computations easily, through custom implementations of corresponding data structures and methods such as matrix computations, as well as solvers and optimisation routines. Ebi as a framework ensures a consistent parameter handling, and is thus well positioned to act as a background library for Pm4py, BupaR and ProM in the future. Ebi is open source and we welcome contributions from the community.

The remainder of this paper is organised as follows. In Section 2, we discuss the significance of Ebi for stochastic process mining. Section 3 details its innovations. Section 4 lists the currently implemented techniques, while Section 5 discusses its maturity and Section 6 concludes the paper.

2. Significance

In stochastic process mining, techniques may need to deal with astonishingly small numbers. For instance, the BPIC11 [4] log has traces with more than 1 800 events. In any reasonable stochastic process model, the probability of such a trace would be well below 10^{-300}, which is the minimum value that a double precision float can represent. Even if we could represent such low values reliably, doing any kind of computation would yield a low precision. For instance, if we want to compute the unit Earth-movers' stochastic conformance measure [5], we need to sum over one such value for each trace in the log, which would, in a typical log with thousands of traces, not leave any precision. In practice, computations on such logs without exact arithmetic would be highly unreliable. More numerically complex techniques that for instance require matrix computations like solving, inversion or multiplication with such low values would fare even worse. Hence, exact computations are a necessity in stochastic process mining, and Ebi is the first process mining framework that provides exact arithmetic for stochastic process mining.

Furthermore, having a command line interface eases the application of stochastic process mining techniques in demanding scientific experimental settings. Ebi is fully command-line based; though integrations with existing process mining frameworks such as Pm4py [1], BupaR [2] and ProM [3] are planned, in a similar fashion to Rust4pm [6].

Finally, Ebi encourages techniques to accept a large variety of inputs and produce a large variety of outputs, as it moves the burden of matching inputs and outputs of techniques to file types in the framework itself: by means of abstract interfaces (such as "a stochastic language

that we can iterate over, and that iteration will end"), file types implement interfaces while techniques use them.

3. Innovations

Ebi aims to provide stochastic process mining techniques to analysts in a consistent and coherent fashion, accessible through the command line. The Ebi framework provides the following innovations:

- **Background library & command line**
  Ebi is a command line tool, which allows it to be applied easily in scientific experiment workflows. However, due to its structured set-up, it is well suited to be used as well as a fast background library for graphical tools or scripting languages such as Pm4py [1], BupaR [2] and ProM [3].
- **Exact computation**
  As described in Section 2, exact computations are a necessity in stochastic process mining. Completely transparent to end users and techniques, Ebi uses fractions of unlimited size, explicit logarithms and explicit roots to perform almost all computations exactly. However, exact computations can be disabled by the user for performance reasons. For more details, please refer to the manual.
  Unfortunately, most existing Rust libraries do not accept unlimited fractions as input, let alone explicit representations of logarithms and roots. Therefore, many secondary techniques, such as linear solvers, matrix computations and approximation techniques, were adapted to support exact computations.
- **Input and output detached from techniques**
  Completely transparent for end users, an implemented technique that needs to e.g. walk over the traces of an event log will automatically support all input formats for which that can be done. That is, techniques preferably define their inputs in terms of interfaces ("traits"). Again completely transparent to end users and techniques, the output of a technique is converted automatically by Ebi based on the file extension of the desired output file. For more details, please refer to the manual.
- **Local and memory-safe**
  Ebi runs locally and neither needs nor uses internet access, which makes it suitable for privacy-sensitive settings. Use of the Rust programming language ensures most types of memory-related errors are absent. Furthermore, Rust is, of course dependent on the programmer, more energy efficient than Java and much more energy efficient than Python [7].

4. Techniques, Files & Architecture

In Ebi, a *command* is the implementation of an algorithm. As summarised in Figure 1, a command defines the inputs it needs, in terms of traits and object types, as well its output type, in terms of an object type it will produce. For instance, the command `Ebi analyse most-likely-traces`

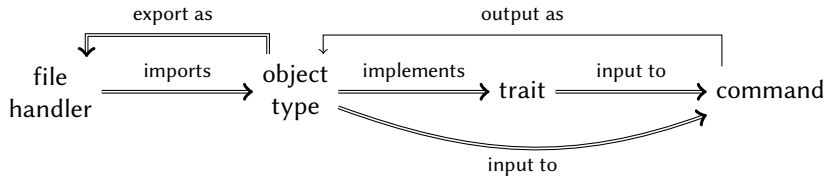

Figure 1: Architecture of Ebi.

Table 1
Files supported by Ebi.

	extension(s)	remarks
event log	.xes, .xes.gz	
finite stochastic language	.slang	
finite language	.lang	
stochastic deterministic finite automaton	.sdfa	
stochastic labelled Petri net	.slpn	
labelled Petri net	.lpn	
accepting Petri net	.pnml	every deadlock is considered a final marking
directly follows model	.dfm	

will extract the most likely traces from its input. These inputs are (i) something that implements the `FiniteStochasticLanguage` or `StochasticDeterministicSemantics` trait, and (ii) an integer. When the user issues a command, Ebi will establish which traits and object types can serve as an input for the given command. For each of these traits and object types, Ebi will search which file handlers can provide it. For our example, the file handlers that can provide a usable input are a finite stochastic language (.slang), a compressed event log (.xes.gz), a stochastic labelled Petri net (.slpn), a stochastic deterministic finite automaton (.sdfa) and an event log (.xes). Ebi will attempt to load the given file with all of these file handlers, until it finds one that parses. Then, the command is executed. The command will result in an output, that is, an object of a particular object type. Then, Ebi will consider the file extension of the file the user wishes to write the result to, and see whether there is a file handler with that extension that can export the output object type.

The main techniques for which Ebi currently provides an implementations are shown in Table 2. In total, Ebi currently has 30 commands. For a full overview of commands and their inputs and outputs, please refer to the manual, accessible through http://ebitools.org.

5. Maturity

Ebi has been used in the experiments described in several recent papers published by the author team [12, 15, 16]. In these experiments, Ebi has shown to be me much faster than corresponding, earlier, implementations in other frameworks. This is presumably partly due to implementing these techniques a second time, though perhaps also due to the lower level access to computing

Table 2
Techniques currently implemented in Ebi.

Technique	command	paper
Return all traces with their probabilities	`Ebi ana all`	
Estimate the completeness of an event log	`Ebi ana comp`	[8]
Obtain the trace(s) that are on average the closest to all traces in an event log	`Ebi ana med`	
Get all traces that have a likelihood higher than a threshold	`Ebi ana minprob`	
Get the most likely traces	`Ebi ana mostlikely`	
Align a finite (non-stochastic) language to a model ↣ computes alignments for any model, not just Petri nets	`Ebi anans ali`	[9]
Cluster traces into most-dissimilar groups (non-stochastic)	`Ebi anans clus`	
Obtain the trace(s) that are on average the closest to all traces in an event log (non-stochastic)	`Ebi anans med`	
Association between process behaviour and trace attributes	`Ebi asso atts`	[10]
Compute entropic relevance (uniform) ↣ computes log-model comparisons for any model, not just SDFAs	`Ebi conf er`	[11]
Compute Jensen-Shannon stochastic conformance	`Ebi conf jssc`	[12]
Compute unit earth-movers' stochastic conformance	`Ebi conf uemsc`	[5]
Convert stochastic languages and stochastic deterministic finite automata into other types	`Ebi conv …`	
Discover a stochastic model using alignments	`Ebi disc ali`	[13]
Discover a stochastic model using occurrences	`Ebi disc occ`	[13]
Discover a stochastic model by giving each transition a weight of 1	`Ebi disc uni`	
Print basic information on any file	`Ebi info`	
Compute the probability that a stochastic model will produce a trace of a specified language	`Ebi prob mod`	[14]
Compute the probability of a trace in a model	`Ebi prob trac`	[14]
Compute the most likely path a trace followed in a model	`Ebi prob exptra`	
Sample a stochastic language	`Ebi sam`	
Test whether the sub-logs defined by a categorical trace attribute are derived from identical processes	`Ebi tst lcat`	[10]
Test-parse a file	`Ebi vali …`	
Visualise as graph	`Ebi vis svg`	

resources that Rust provides; something also observed in the Rust4pm project [6]. Furthermore, Ebi has proven to be much easier to integrate in a scientific experimental setting, due to its command-line interface: a simple command-line script suffices to perform complex chains of tasks, whereas in ProM or Pm4py, all file IO and intermediate-result storage, and all exceptions, need to be programmed. Finally, Ebi has been applied in several case studies with industry partners, including amongst other things in thesis projects.

6. Conclusion

Ebi is a framework for stochastic process mining. It offers exact arithmetic computation and, currently, over 30 algorithms related to stochastic process mining, ranging from completeness estimation to analysis, conformance checking, discovery, visualisation and statistical tests. For technique implementers, Ebi handles the input and output file handling: techniques only need to define their input and output format, and Ebi will take care of transformations automatically. Currently, Ebi is a command-line based tool, however, in future work, it is intended that Ebi can serve as a high speed background library for existing process mining frameworks such as BupaR, ProM and Pm4py.

References

[1] A. Berti, S. J. van Zelst, W. M. P. van der Aalst, Process mining for python (pm4py): Bridging the gap between process- and data science, CoRR abs/1905.06169 (2019).

[2] G. Janssenswillen, B. Depaire, M. Swennen, M. Jans, K. Vanhoof, bupar: Enabling reproducible business process analysis, Knowl. Based Syst. 163 (2019) 927–930.

[3] B. F. van Dongen, et al., The prom framework: A new era in process mining tool support, in: ICATPN, volume 3536 of *LNCS*, Springer, 2005, pp. 444–454.

[4] B. van Dongen, Real-life event logs - hospital log, 2011.

[5] S. J. J. Leemans, A. F. Syring, W. M. P. van der Aalst, Earth movers' stochastic conformance checking, in: BPM Forum, volume 360 of *LNBIP*, Springer, 2019, pp. 127–143.

[6] A. Küsters, W. M. P. van der Aalst, Rust4pm: A versatile process mining library for when performance matters, in: BPM Demos, volume to appear, CEUR-WS.org, 2024, p. to appear.

[7] R. Pereira, M. Couto, F. Ribeiro, R. Rua, J. Cunha, J. P. Fernandes, J. Saraiva, Ranking programming languages by energy efficiency, Sci. Comput. Program. 205 (2021) 102609.

[8] M. Kabierski, M. Richter, M. Weidlich, Addressing the log representativeness problem using species discovery, in: ICPM, IEEE, 2023, pp. 65–72.

[9] A. Adriansyah, B. F. van Dongen, W. M. P. van der Aalst, Conformance checking using cost-based fitness analysis, in: EDOC, IEEE Computer Society, 2011, pp. 55–64.

[10] S. J. J. Leemans, J. M. McGree, A. Polyvyanyy, A. H. M. ter Hofstede, Statistical tests and association measures for business processes, IEEE TKDE 35 (2023) 7497–7511.

[11] H. Alkhammash, et al., Entropic relevance: A mechanism for measuring stochastic process models discovered from event data, Inf. Syst. 107 (2022) 101922.

[12] T. Li, S. J. J. Leemans, A. Polyvyanyy, The jensen-shannon distance metric for stochastic conformance checking, in: ICPM workshops, volume to appear of *LNBIP*, 2024.

[13] A. Burke, S. J. J. Leemans, M. T. Wynn, Stochastic process discovery by weight estimation, in: ICPM Workshops, volume 406 of *LNBIP*, Springer, 2020, pp. 260–272.

[14] S. J. J. Leemans, F. M. Maggi, M. Montali, Enjoy the silence: Analysis of stochastic petri nets with silent transitions, Inf. Syst. 124 (2024) 102383.

[15] S. J. J. Leemans, T. Li, M. Montali, A. Polyvyanyy, Stochastic process discovery: Can it be done optimally?, in: CAiSE, volume 14663 of *LNCS*, Springer, 2024, pp. 36–52.

[16] W. M. van der Aalst, S. J. J. Leemans, Learning generalized stochastic petri nets from event data, in: Festschrift Joost-Pieter Katoen, volume to appear of *LNCS*, 2024.