

PURPLE: a PURPose-guided Log GEnerator (Extended Abstract)

Andrea Burattin¹, Barbara Re², Lorenzo Rossi^{2,*} and Francesco Tiezzi³

¹Dpt. of Applied Mathematics and Computer Science, Technical University of Denmark, Kgs. Lyngby, Denmark

²School of Science and Technology, University of Camerino, Camerino, Italy

³Dpt. of Statistica, Informatica, Applicazioni, University of Florence, Florence, Italy

Abstract

Process mining collects a variety of techniques. To test and compare these techniques, we need event logs tailored to their specific mining purposes, e.g., process discovery and conformance checking. To this aim, we propose the PURPLE tool, a generator of event logs supporting different mining purposes. PURPLE performs guided simulations of a business model, shaping the resulting event log by the selected mining purpose.

Keywords

Process mining, Event log, Simulation

1. Introduction

Process mining techniques play a crucial role in extracting non-trivial information from the execution data of business processes. The effectiveness and the precision of process mining depend on the reliability of the underlying algorithms, whose development requires testing them against event logs that suit the purpose for which the algorithm has been devised [1, 2]. Obtaining event logs fitting a specific purpose is a complex yet necessary achievement since bad quality logs hamper the use of process mining [3]. In this regard, several approaches, e.g., [4, 5], propose the automated generation of artificial event logs via the simulation of models in a predetermined language. However, most of them are not meant to produce logs fulfilling some specific properties. Instead, they are purpose-agnostic i.e., simulate random execution traces producing a different event log every time.

To address this problem, we propose PURPLE: a PURPose-guided Log gEneration tool that implements the homonym framework we introduced in [6]. PURPLE is a web application able to simulate BPMN[7] and Petri-net[8] models and produce event logs tailored to process discovery and conformance checking purposes. Its key feature lies in the architecture, which can be adapted to many process modeling languages and mining purposes. The tool performs *guided*

ICPM 2022 Doctoral Consortium and Tool Demonstration Track


*Main contributor and corresponding author.

✉ andbur@dtu.dk (A. Burattin); barbara.re@unicam.it (B. Re); lorenzo.rossi@unicam.it (L. Rossi); francesco.tiezzi@unifi.it (F. Tiezzi)

🆔 0000-0002-0837-0183 (A. Burattin); 0000-0001-5374-2364 (B. Re); 0000-0002-6872-0616 (L. Rossi); 0000-0003-4740-7521 (F. Tiezzi)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

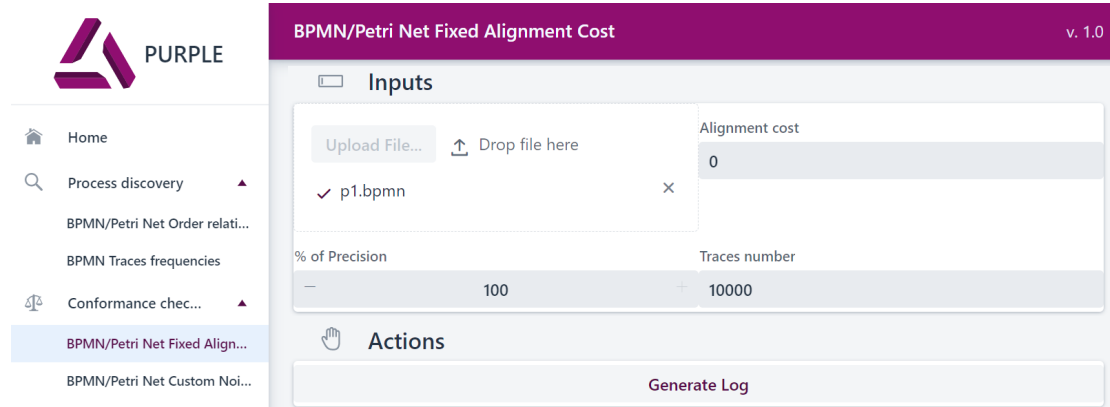


Figure 1: PURPLE interface.

simulations of an input process model to incrementally generate specific execution traces until the produced log satisfies the desired mining purpose. We assess the maturity of PURPLE through thought experiments where we measure the quality of logs generated by PURPLE.

The rest of the paper is structured as follows. Section 2 presents the PURPLE tool and its features, then Section 3 discusses the tool maturity. Finally, Section 4 indicates where to download and access the tool.

2. PURPLE main features

The PURPLE tool, Figure 1 is a progressive web app that can be used online as a service or executable on a local machine. It implements the PURPLE framework described in [6] thus, differently from tools that generate logs from random simulators of process models, it produces from guided simulations event logs tailored to the mining purpose under investigation. The key innovation of PURPLE lies in its architecture grounded on three components: a **semantic engine**, an **evaluator**, and a **simulator**. Except for the simulator that is fixed, the tool can be instantiated with a semantic engine supporting a given modeling language (e.g., BPMN, Petri Net), and an evaluator tailored to a desired mining purpose (e.g., process discovery, compliance checking). The **semantic engine** implements the semantics of the input model. Given an execution status, it returns the set of reachable states (thus the following activities that the model can perform), hence it lets the tool generate a labeled transition system (LTS) of the input model. The **evaluator** checks ‘how much’ an event log satisfies the properties needed for the purpose under consideration. The result of the evaluation is the *delta* that will drive the simulator. A delta is a sub-trace that acts as a bias for the simulation indicating the parts of the LTS to be traversed, thus influencing the produced traces. The **simulator** explores the LTS to produce execution traces to be added to the log. By taking as input the *delta*, the simulator tries to include the suggested sub-trace in the LTS traversal. This guarantees the production of traces, and hence a log, that satisfies the desired mining purpose.

By fixing a modeling language and a mining purpose, we get an instantiation of PURPLE ready for producing logs. The event log is created in iterative steps. In each of them, the tool adds a new trace to the log and checks if the purpose is satisfied or not. If not, the next step is

guided to produce a new trace that better shapes the log for the target purpose.

Besides implementing the architecture, the PURPLE tool provides different instantiations of the framework. It implements four evaluators addressing mining purposes about process discovery and conformance checking, and two semantic engines permitting to simulate BPMN and Petri-net modes. In the following, we describe these instantiations from the evaluator's point of view since it is devised to shape the produced log.

The first instantiation, **process discovery via order relations**, is devised for algorithms relying on the order relation between activities to obtain an accurate version of the original model, like the Alpha miner. This instantiation generates event logs covering the order relations seen in the input model without logging multiple times the same trace hence, the same order relations. Therefore, PURPLE produces the smallest log covering the relations in the footprint matrix. The evaluator component implemented for this purpose compares the order relations found in the input model with the ones generated in the log up to that moment, producing a delta accordingly. For instance, if the log misses a sequence relation between activities A and B the delta will contain the sub-trace $\langle A, B \rangle$. In such a way, the next simulation tries to produce a trace containing $\langle A, B \rangle$. Once all the order relations of the input model are covered, PURPLE produces as output the *.xes* file.

The second instantiation, **process discovery via frequencies**, aims at generating event logs for discovery algorithms based on frequencies, such as the Heuristics miner. To address this purpose, it produces logs where some traces could be less or more frequent than others. During the log generation, the evaluator calculates in the current log the occurrences of traces and the thresholds for the loops chosen by the user, then generates a delta accordingly. If some of these values are lower than requested, the evaluator passes a delta to the simulator containing the entire traces still infrequent in the log. Once the requested occurrence percentages are satisfied, PURPLE returns the log *.xes* file.

The other two instantiations regard conformance checking. To check the reliability of such techniques or to compare their performances, it is necessary to have logs embedding traces with deviations from the normal behavior, i.e., noisy behaviors. We propose two instantiations of PURPLE producing event logs from BPMN and Petri-nets with a precise amount of noisy behavior or with a precise alignment cost.

The **conformance checking via noise frequencies** instantiation generates event logs with the desired percentages of noisy traces. The user can choose the noise percentages for *missing head*, *missing tail*, *missing episode*, *order perturbation*, and *additional event*. In this case, the evaluator passes an empty delta to the simulator to return a random trace without noise. Then, it compares the percentage of occurrences for each type of noise in the current log for the requested one. Consequently, the trace is modified introducing the type of noise farthest from the requested occurrence. Once PURPLE reaches the desired percentages, it returns the log.

The last instantiation, **conformance checking via fixed align cost**, generates event logs with a precise amount of noise that involves a specific cost for the alignment. For this purpose, the tool extracts from the model the set of traces that can be produced and uses them later for calculating the alignment costs. Then, similarly to the purpose above, the evaluator receives from the simulator traces without noise, perturbs them with a type of noise, and updates the reached alignment cost. Every time a noisy trace is added to the current log, the evaluator calculates the optimal alignment cost between the noisy trace and traces previously extracted

from the model.

3. Tool maturity

To show the maturity, we provide the results of experiments where we measured the quality of the event logs produced by each PURPLE instantiation. When possible, we compare our results with the ones achieved by reference tools, such as PLG2, BIMP (<https://bimp.cs.ut.ee/>), and the ProM plugin of the GED methodology (<https://www.promtools.org/>). For the experiments, we used BPMN and Petri-net models whose dimension ranges from 8 to 53 elements. They are both structured and unstructured, and some of them contain loops.

Regarding the process discovery via order relations, PURPLE covers for all the models the 100% of relations, while PLG2, GED, and BIMP collected 93.1%, 77.1%, and 44.1% of relations respectively. In the process discovery via frequencies, PURPLE always produces logs with the required number of traces per type. In this case, PURPLE was not compared with any other tool since neither PLG2, GED, nor BIMP produces logs based on frequencies. In the conformance via noise frequencies, PURPLE produces exactly the required frequencies of noised traces, while PLG2, used for the comparison, approximates this result with an error of 20.8%. Finally, in the conformance checking with fixed align cost, PURPLE produces logs with alignment costs that, in the worst case, differ only from the 2.3% for the required cost. During the experiments, we measured the time spent by each tool for generating the logs. The reference tools take on average about 5 seconds, while PURPLE around 15 seconds, depending on the purpose. Even if PURPLE is less efficient than the other tools, it produces better logs in a reasonable amount of time.

4. Screencast and Website

The PURPLE tool, as well as its source code, examples, artifacts generated during the experiments, user guide, and screencast, are available at <https://pros.unicam.it/purple/>.

References

- [1] B. Van Dongen, A. De Medeiros, L. Wen, Process mining: Overview and outlook of petri net discovery algorithms, in: *ToPNoC*, volume 5460 of *LNCS*, Springer, 2009, pp. 225–242.
- [2] A. de Medeiros, C. Günther, Process Mining: Using CPN Tools to Create Test Logs for Mining Algorithms, in: *Practical Use of Coloured Petri Nets and CPN Tools*, volume 576, 2005, pp. 177–190.
- [3] R. Bose, R. Mans, W. van der Aalst, Wanna improve process mining results?, in: *CIDM*, IEEE, 2013, pp. 127–134.
- [4] A. Burattin, PLG2: Multiperspective Process Randomization with Online and Offline Simulations, in: *BPM Demo Track*, volume 1789, CEUR-WS.org, 2016, pp. 1–6.
- [5] E. Esgin, P. Karagoz, Process Profiling based Synthetic Event Log Generation, in: *IC3K*, volume 1, SCITEPRESS, 2019, pp. 516–524.

- [6] A. Burattin, B. Re, L. Rossi, F. Tiezzi, A Purpose-Guided Log Generation Framework, in: Business Process Management, volume 13420 of *LNCS*, 2022, pp. 181–198.
- [7] F. Corradini, C. Muzi, B. Re, L. Rossi, F. Tiezzi, Formalising and animating multiple instances in BPMN collaborations, *Information Systems* 103 (2022).
- [8] J. Meseguer, U. Montanari, V. Sassonet, On the semantics of Petri nets, in: *CONCUR*, volume 630 of *LNCS*, Springer, 1992, pp. 286–301.