IM-viz: A Tool for the Step-by-step Visualization of the Inductive Miner

Florian Lang*, Din Hida*, Yingjie Bian*, Adrian Rebmann*, Han van der Aa*
*School of Business Informatics and Mathematics, University of Mannheim, Germany
Email: {florian.vincent.lang | din.hida | yingjie.bian}@students.uni-mannheim.de
{rebmann | han.van.der.aa}@uni-mannheim.de

***Abstract*—The *Inductive Miner* is a state-of-the-art algorithm for process discovery and a staple in process mining education, since its divide-and-conquer nature can teach students how to recognize behavioral relations in event data and to break up a discovery problem into smaller parts. However, a key problem from this educational perspective is that the algorithm's manual application is time consuming, involving a considerable amount of drawing, whereas existing implementations of the algorithm only show the final outcome, not its intermediary steps. To overcome this, we present *IM-viz*, an educational process mining tool that visualizes the application of the *Inductive Miner* and the *Inductive Miner infrequent* in an iterative manner. *IM-viz* allows users to interactively explore how inductive mining works and how it deals with different kinds of event data, thus providing a convenient means for process mining students and educators to establish and analyze step-by-step process discovery examples.**

***Index Terms*—process mining, inductive miner, algorithm visualization**

I. Introduction

Process discovery is a quintessential task in process mining, which takes exemplary process executions, stored in an event log, and aims to establish a process model that accurately describes the underlying process [1]. In this context, inductive mining refers to a family of discovery algorithms that work in a top-down manner [2]. These algorithms apply a divide-and-conquer strategy to recursively decompose the process discovery task into smaller parts. This is achieved by establishing a directly-follows graph (DFG) of the event log and, subsequently, identifying *cuts* that partition the activities in the DFG into smaller sets. These cuts identify behavioral relations between the activity sets, indicating that they are in a sequential, parallel, exclusive, or loop relation. The algorithm is recursively applied on the sub logs corresponding to these smaller activity sets, until the problem cannot be further decomposed and a process tree or workflow net is returned.

Inductive mining is recognized as (part of the) state of the art for process discovery, as it provides formal guarantees on its output (most importantly, output models are always sound), is scalable, and provides flexibility to users, since inductive mining algorithms exist that can deal with different quality issues, such as noise and incompleteness. Furthermore, inductive mining is interesting from an educational perspective, as its divide-and-conquer nature provides a convenient means for students to learn how to break up a discovery task into smaller problems and to recognize behavioral relations in event data.

A downside from this educational perspective is that students new to process mining may find it difficult to understand how the algorithms exactly work and how they should be applied. Additionally, the algorithms are time consuming to apply in a manual manner, involving a considerable amount of drawing (of the DFGs). Since existing implementations only provide the output obtained by discovery (i.e., the final process tree or workflow net), these do not allow users to understand the intermediary steps taken to reach this result. Consequently, it is tedious for educators and difficult for students to gain insights into the application of inductive mining algorithms in a step-by-step manner.

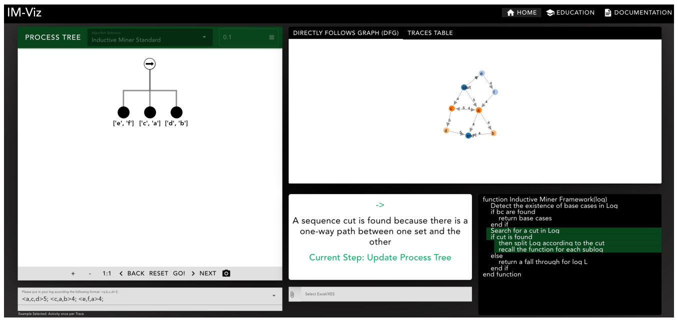
To alleviate this issue, we present IM-viz, an educational process mining tool that visualizes the application of the standard *Inductive Miner* and the *Inductive Miner infrequent* algorithms in a step-by-step manner, showing the intermediary steps taken to go from input to output. The creation of IM-viz is inspired by the *VisuAlgo* project [3] of the National University of Singapore (NUS), which visualizes popular algorithms and data structures in an intuitive manner.

II. The IM-viz Tool

At https://github.com/badrecursionbrb/im-viz IM-viz is available. This repository provides a link to a deployed instance of IM-viz, the source code and instructions to run the tool locally, documentation, and a video showing its usage.

A. Visualizer

The main page of the IM-viz application, directly accessed when opening it, focuses on the visualization of the inductive mining algorithms, as shown in Figure 1.

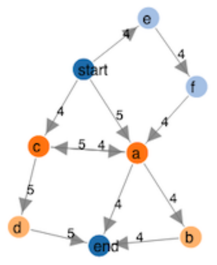


Please put in your log according the following format: <a,b,c,d>3;
<a,c,d>5; <c,a,b>4; <e,f,a>4;
Example Selected: Activity once per Trace

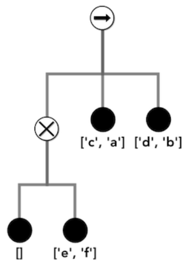
->
A sequence cut is found because there is a one-way path between one set and the other
Current Step: Update Process Tree

```
function Inductive Miner Framework(log)
    Detect the existence of base cases in Log
    if bc are found
        return base cases
    end if
    Search for a cut in Log
    if cut is found
        then split Log according to the cut
        recall the function for each sublog
    else
        return a fall through for log L
    end if
end function
```

Fig. 1: Screenshot of the main part of the *IM-viz* tool, the visualizer. Depicted is an example with the first step

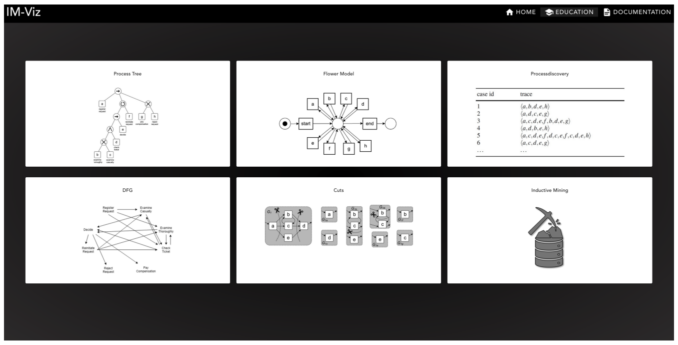


(a) DFG visualization, each color indicates an activity set identified by a cut



(b) Process tree under construction, brackets indicating activity sets to be explored.

Fig. 2: Screenshots of the main graphs in the visualizer



case id	trace
1	$\langle a,b,d,e,h\rangle$
2	$\langle a,d,c,e,g\rangle$
3	$\langle a,c,d,e,f,b,d,e,g\rangle$
4	$\langle a,d,b,e,h\rangle$
5	$\langle a,c,d,e,f,d,c,e,f,c,d,e,h\rangle$
6	$\langle a,c,d,e,g\rangle$
…	…

Fig. 3: Screenshot from the front page of the education section

Getting started. To start visualizing a discovery task, users first select the event data and algorithm to apply. In terms of event data, IM-viz allows users to upload an XES file, simply write down an event log using a string representation (e.g., “<a,c,d>4;<c,a,a,d>5;”), or select one of the ready-made examples. In terms of algorithms, users can currently select the standard *Inductive Miner* and the *Inductive Miner infrequent*. If the *Inductive Miner infrequent* is selected, one needs to additionally enter the desired noise threshold. After that, the user clicks the “Go!" button to start the selected algorithm.

Algorithm visualization. When a user presses “Go!" (or, after the first step, “Next”), IM-viz will apply one step of the selected discovery algorithm on the selected event data. As shown in Figure 1, the tool visualizes the application of this step through three complementary components:

1. The top-right window shows the DFG of the (sub-) log that is currently being considered. The DFG visualization, depicted in Figure 2a, uses different colors to indicate different activity sets that will be separated by the cut to be performed.

2. The bottom-right part of the screen shows the inductive mining algorithm that is currently being applied and which lines are currently being executed, showing if the algorithm is currently in a base case, if a cut has been identified, or if a fall-through scenario is reached (where no cut can be applied). The box next to this algorithm, then, provides more information on this current step, e.g., by indicating which cut has been found and why this cut can be applied to the DFG at hand.

3. The left-hand side of the screen shows the current state of the process tree, which is updated after each step. The tree representation, depicted in more detail in Figure 2b, uses, e.g., [’e, f’] to indicate an activity set that needs further exploration. After updating each of these components in a sequential manner, the application of the algorithm is paused, so that users can take the time to explore the current, intermediary state of the discovery task. When ready, users can press “Next” to apply the next step, which corresponds to the left-most node in the tree that requires further exploration.

B. Education Section

IM-viz also includes an education section (accessible via the top-right corner of the UI). It provides information on the most important concepts necessary for a general understanding of the *Inductive Miner* were selected and explained briefly such as process trees or the so called *flower model*. A screenshots of the education section landing page is shown in Figure 3.

C. Tool Architecture

The IM-viz application consists of a Python-based back-end and a JavaScript-based front-end, which communicate via REST requests. The back-end uses *Flask* to implement the REST-API and *PM4Py* [4] as a basis for the *Inductive Miner* algorithm implementation. Because accessing intermediate results required code changes, *PM4Py* was forked to keep the distinction between IM-viz and the original *PM4Py* framework. The front-end is built with *vue.js* and *node.js*. For visualizations, we used the *Data Driven Documents (D3)* library, together with the *Cola.js* implementation for the network graph. The front-end can be hosted by any HTTP server.

III. Conclusion and Future Work

IM-viz provides students with step-by-step guidance through the application of inductive mining algorithms using event data of their choice. The tool provides different kinds of information for each step of an algorithm, showing the current step in the algorithm itself, the DFG, and the process tree, allowing users to jump back and forth between steps.

As next steps, we aim to extend the tool to includes other process mining algorithms, such as the alpha algorithm for process discovery or the A^* search in alignment-based conformance checking. Additionally, the visualizations itself may be improved in terms of comprehensibility and vividness. Finally, we aim to conduct user studies to assess the benefits of using algorithm visualization in process mining education.

References

[1] W. M. van der Aalst and J. Carmona, *Process mining handbook*. Springer Nature, 2022.

[2] S. J. J. Leemans, “Robust Process Mining with Guarantees: Process Discovery, Conformance Checking and Enhancement,” Ph.D. dissertation, Eindhoven University of Technology, Eindhoven, 2017.

[3] S. Halim, F. Halim, K. Z. Chun, V. Loh Bo Huai, P. Thi Quynh Trang, P. Phandi, A. Millardo tijndradinata, N. Hoang Duy, R. M. Tan Zhao Yun, and I. Reinaldo, “VisuAlgo - visualising data structures and algorithms through animation,” 2011. [Online]. Available: https://visualgo.net/en

[4] A. Berti, S. van Zelst, and W. M. van der Aalst, “Process Mining for Python (PM4Py): Bridging the Gap Between Process- and Data Science,” in *ICPM Demos 2019*, vol. 2374. CEUR, 2019, pp. 13–16.