



University of Camerino

SCUOLA DI SCIENZE E TECNOLOGIE

Computer Science course (LM-18)

Process Mining Project

Student
Kevin Concettoni

Student ID 131994

Professor
Barbara Re

A.A. 2024/2025

Contents

1	Introduction	9
2	Dataset	11
3	Tools and experiments	13
3.1	Disco	13
3.2	Hospital Billing with Disco	14
3.3	Pm4py	15
3.3.1	Alpha miner	16
3.3.2	Heuristic miner	16
4	Conclusions	19

Listings

3.1	Pytohn code to run the alpha algorithm with pm4py	16
3.2	Python code to run the heuristic algorithm with pm4py	17

List of Figures

2.1	Example of logs of Hospital Billing dataset	11
3.1	Automated process discovery with absolute frequency	14
3.2	Automated process discovery with total duration	15
3.3	Fitness of alpha algorithm	16
3.4	Fitness of heuristic algorithm	17
3.5	Petri net derived from Heuristic algorithm	17

1. Introduction

Process mining is an emerging discipline within computer science that bridges the gap between data science and business process management. It focuses on analyzing and improving real-life business processes by extracting insights from event logs recorded by information systems. With the growing complexity of organizational operations and the abundance of data generated, process mining has become a vital tool for organizations seeking to gain visibility into their internal workflows, identify inefficiencies, and support data-driven decision-making.

This project is centered on exploring, analyzing, and comparing different process mining platforms to evaluate their capabilities, performance, and usability. The core objective is to assess how various tools interpret and visualize event logs, as well as how effectively they support process analysis tasks such as process discovery, conformance checking, and performance analysis.

To achieve this, I utilized two widely recognized process mining tools: Disco and PM4PY. By conducting a series of experiments on a standardized event log dataset in XES format, I examined how each tool processes the data, the types of process models they generate, and the features they offer for analysis. Multiple versions of the XES files were also created and tested to evaluate how each platform handles different data variations and scenarios.

Through this comparative study, the project aims to provide an in-depth understanding of the strengths and limitations of Disco and PM4PY. The ultimate goal is to offer practical insights that can assist researchers, analysts, and businesses in selecting and utilizing the most appropriate process mining tools based on their specific requirements and datasets.

2. Dataset

The Hospital Billing Event Log [Man17], as presented by Mannhardt (2017), originates from the financial modules of the Enterprise Resource Planning (ERP) system utilized by a regional hospital. This event log provides a detailed record of activities associated with the billing processes for medical services rendered by the hospital. Each individual trace within the log corresponds to a complete sequence of billing activities executed for a bundled package of medical services, rather than capturing the clinical or medical procedures themselves. This distinction is critical, as the event log exclusively reflects administrative and financial process steps related to invoicing and payment, omitting direct information about patient treatment or clinical interventions.

The dataset comprises approximately 100,000 traces, which represent a randomly selected subset of process instances collected over a period of three years. This extensive temporal coverage ensures that the data encapsulates a wide variety of billing scenarios, seasonal effects, and process variations that may arise in a real-world healthcare environment. Each trace is enriched with multiple attributes, including but not limited to the current state of the process, the type of case being handled, and underlying diagnostic information. To preserve confidentiality and comply with privacy regulations, all sensitive data within the event log has been anonymized rigorously. Specifically, attribute values have been anonymized and timestamps associated with events have been deliberately randomized. Importantly, while the absolute event times have been altered, the relative timing between consecutive events within each trace has been maintained intact, thereby preserving the process temporal dynamics.

This anonymized, large-scale event log offers a valuable resource for the process mining community, enabling in-depth analysis of billing workflows, identification of bottlenecks, deviations, and opportunities for process improvement in the healthcare billing domain. Its combination of realistic complexity, substantial volume, and temporal fidelity makes it particularly suited for evaluating and benchmarking process mining algorithms and tools. Consequently, the Hospital Billing event log serves as a critical dataset for advancing research on the automation and optimization of healthcare financial processes.

Case ID	Events	Variant	Started	Finished	Duration
A	5	Variant 3	16.12.2012 19:33:10	19.12.2013 03:44:31	1 year, 2 days
B	2	Variant 4	16.12.2012 19:33:50	19.10.2013 12:37:05	306 days, 16 hours
C	10	Variant 130	13.01.2013 21:04:24	23.05.2013 07:32:15	129 days, 9 hours
D	5	Variant 3	16.01.2013 21:05:37	09.02.2014 21:56:28	1 year, 24 days
E	6	Variant 1	18.01.2013 09:07:53	21.05.2013 09:50:35	122 days, 23 hours
F	6	Variant 1	10.01.2013 08:41:37	04.08.2013 22:46:17	206 days, 13 hours
G	6	Variant 1	18.12.2012 09:07:50	12.07.2013 22:16:45	206 days, 12 hours
H	6	Variant 1	22.01.2013 09:35:15	03.08.2013 19:03:37	193 days, 8 hours
I	7	Variant 6	24.12.2012 07:34:53	03.04.2013 23:34:05	100 days, 14 hours
J	6	Variant 1	20.12.2012 09:05:53	10.03.2013 04:22:05	79 days, 19 hours
K	6	Variant 1	18.12.2012 09:10:30	29.06.2013 19:23:16	193 days, 9 hours
L	2	Variant 4	23.12.2012 10:33:42	23.12.2012 10:34:04	22 secs
M	6	Variant 1	25.12.2012 10:36:18	12.04.2013 10:52:53	107 days, 23 hours

Figure 2.1: Example of logs of Hospital Billing dataset

3. Tools and experiments

This chapter presents the technologies and tools employed to analyze and compare the results derived from a single event log extracted from the selected dataset. The objective is to evaluate how different process mining platforms interpret and visualize the same data. Some of the tools utilize well-established process discovery algorithms, such as the Alpha Miner and Heuristic Miner—both of which were discussed during the course—while others adopt alternative approaches or proprietary algorithms for process model generation. Each tool will be introduced and examined individually, with particular attention to its analytical capabilities, strengths, and limitations in handling the dataset under study.

3.1 Disco

Disco is a sophisticated and widely adopted process mining application developed to support the in-depth analysis of business process event logs. Its primary objective is to transform raw event data into insightful and interactive process models through automated process discovery techniques. Unlike tools that strictly implement formal algorithms like Alpha Miner or Heuristic Miner, Disco utilizes a proprietary fuzzy miner algorithm that balances precision and simplicity, enabling users to extract highly readable and realistic process maps from complex data. One of Disco's most distinctive features is its intuitively understandable process visualization: process maps are automatically generated upon importing a log file, with the thickness of paths indicating the frequency of transitions and the coloration of activities reflecting their relative performance or occurrence. This visual approach enables users to rapidly identify key process flows, bottlenecks, rework loops, and deviations from expected behavior.

The user experience in Disco is streamlined and accessible, offering a clear interface that supports non-technical users while still providing sufficient depth for expert-level analysis. Once a log file—such as an XES-formatted event log—is imported, Disco immediately generates a process map in the "Map" view, allowing the user to interact with the process structure dynamically. Various perspectives can be toggled, including views based on absolute frequency, case frequency, and performance metrics such as throughput time. Users can also filter the number of activities displayed to reduce complexity and improve clarity, especially in large or highly variable processes. Furthermore, Disco provides an analytical view of individual cases and their corresponding variants, making it possible to explore the structure and behavior of each trace within the process in detail. The software supports the analysis of large-scale datasets and is well-suited for organizations of all sizes and sectors that aim to gain actionable insights into their operational workflows.

The second image, instead, represents the performance view of the hospital billing process visualized through the Disco process mining tool. Unlike the previous diagram that showed absolute frequency, this version focuses on duration and timing metrics, showing how long it takes for transitions between activities.

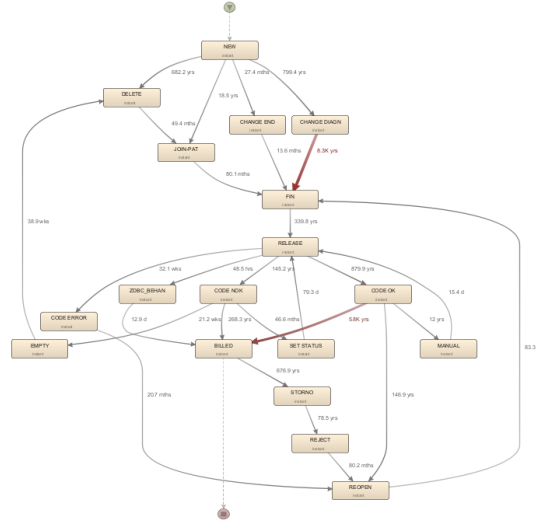


Figure 3.2: Automated process discovery with total duration

This process map reveals extreme variations in performance across transitions, many of which are implausibly high, likely due to anonymization or log errors. Still, Disco helps visualize these trends, enabling analysts to identify paths that require further investigation—either due to real-world inefficiencies or data quality issues. Understanding and addressing these discrepancies is essential for producing reliable models and actionable insights in process mining.

3.3 Pm4py

PM4Py (Process Mining for Python) is a comprehensive open-source library that supports a wide range of process mining techniques within the Python ecosystem. Introduced by Berti et al. (2019), PM4Py was developed to bridge the gap between process science and data science by enabling the analysis of business processes through event logs in a reproducible, extensible, and data-driven manner. The library implements key functionalities such as process discovery, conformance checking, and performance analysis using state-of-the-art algorithms including the Inductive Miner and Alpha Miner. It supports standard event log formats such as XES and CSV, and offers visual and analytical tools to inspect process models like Petri nets and BPMN diagrams. PM4Py’s integration with Python’s data science stack (e.g., pandas, matplotlib) makes it particularly attractive for researchers and practitioners seeking to embed process mining into broader analytics workflows. Its adoption has enabled greater accessibility to process mining techniques and facilitated the development of advanced applications across domains such as healthcare, manufacturing, and auditing [BZA19].

3.3.1 Alpha miner


The alpha algorithm is a process discovery technique used in process mining. It constructs a Petri net model from an event log, aiming to capture the causal dependencies between activities. The algorithm identifies direct succession, causality, and concurrency relationships by analyzing the order of activities in the log. While foundational and relatively simple to implement, the alpha algorithm has limitations, particularly with complex process patterns such as short-loops, non-free-choice constructs, and duplicate tasks, which can lead to incomplete or incorrect process model [Aal11].

In this paragraph, we evaluate the quality of a process model discovered using the Alpha Miner algorithm by conducting a conformance checking procedure based on token replay. First, the Alpha Miner is applied to the event log to derive a Petri net model (net_alpha) along with its initial and final markings. Subsequently, the token_replay method is used to simulate the execution of traces from the event log on the generated model, resulting in a collection of replayed traces. These replay results are then assessed using the replay_fitness_evaluator, which computes fitness metrics that quantify how well the discovered model aligns with the observed behavior in the log. Finally, the overall log fitness score is printed, providing an interpretable metric (between 0 and 1) indicating the proportion of correctly reproduced behavior. This approach enables a quantitative comparison of the discovered model's fidelity with respect to the original event data.

```
net_alpha, im_alpha, fm_alpha = alpha_miner.apply(log)
replayed_traces_alpha = token_replay.apply(log, net_alpha, im_alpha,
    fm_alpha)
fitness_alpha = replay_fitness_evaluator.evaluate(replayed_traces_alpha)
print(f"Alpha Miner Fitness: {fitness_alpha['log_fitness']:.4f}")
```

Listing 3.1: Python code to run the alpha algorithm with pm4py

Here we can see the output after applying the alpha algorithm that derive a fitness around 67%.



```
=== ALPHA MINER ===
replaying log with TBR, completed traces :: 100%| 1020/1020 [00:01:00:00, 628.32it/s]
Alpha Miner Fitness: 0.6798
```

Figure 3.3: Fitness of alpha algorithm

3.3.2 Heuristic miner

Heuristic algorithms, in contrast to exact algorithms, aim to find good-enough solutions to computational problems in a reasonable amount of time, especially when finding optimal solutions is computationally infeasible. These algorithms employ practical methods or shortcuts to navigate complex search spaces, often relying on rules of thumb or informed guesses. In the context of process mining, heuristic algorithms are frequently used to discover process models from event logs by identifying the most significant causal relationships while filtering out infrequent or noisy behavior [WAW06].

This makes them particularly robust to incomplete or imperfect data, providing a balance between model precision and computational efficiency. In this paragraph, we

analyze the structure of a process model discovered using the Heuristics Miner algorithm. The event log is passed to the `discover_petri_net_heuristics` method from PM4Py, which returns a Petri net model (`net_heuristics`) along with its initial and final markings (`im_heuristics` and `fm_heuristics`). We then examine the model's structural complexity by printing the total number of places and transitions, which indicates the size and intricacy of the generated process model. To gain further insight, each place in the model is enumerated and printed with an index, allowing a clear view of the model's state-space. Similarly, each transition is listed, distinguishing between visible transitions (which correspond to observable activities) and silent transitions (which typically represent internal routing logic without direct real-world correspondence). This step provides a detailed look at the internal composition of the heuristically discovered process model and aids in assessing its interpretability and complexity.

```
net_heuristics, im_heuristics, fm_heuristics = pm4py.  
    discover_petri_net_heuristics(log)  
print(f"Heuristics Net - Places: {len(net_heuristics.places)}, Transitions:  
    {len(net_heuristics.transitions)}")  
  
print("Heuristics Net Places:")  
for i, place in enumerate(net_heuristics.places):  
    print(f" {i+1}. {place}")  
  
print("Heuristics Net Transitions:")  
for i, trans in enumerate(net_heuristics.transitions):  
    if trans.label: # Only visible transitions  
        print(f" {i+1}. {trans.label}")  
    else:  
        print(f" {i+1}. [Silent transition]")
```

Listing 3.2: Python code to run the heuristic algorithm with pm4py

Also in this case, we apply the token replay for the conformance checking where we reach a fitness of 91%.

```
Calculating Heuristics Miner fitness...  
replaying log with TBR, completed traces :: 100%  
Heuristics Miner Fitness: 0.9174
```

Figure 3.4: Fitness of heuristic algorithm

As said, here also the petri net derived from the algorithm.

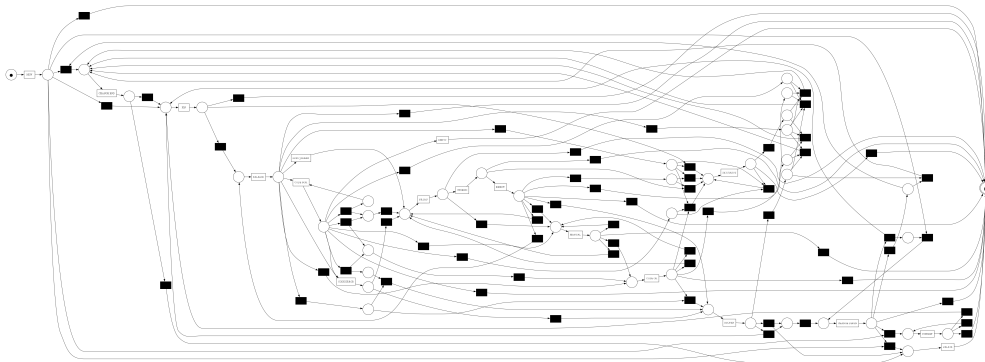


Figure 3.5: Petri net derived from Heuristic algorithm

4. Conclusions

In conclusion, this project critically evaluated and compared two prominent process mining tools, Disco and PM4Py, using the extensive Hospital Billing event log. Our findings reiterate that Disco stands out for its user-centric design, offering an intuitive interface and visually rich process maps that facilitate rapid comprehension and preliminary insights. Its proprietary fuzzy miner algorithm effectively balances precision with simplicity, making it highly accessible for users prioritizing quick visualizations and an easy learning curve.

Conversely, PM4Py emerged as a powerful open-source library within the Python ecosystem, catering to advanced users and researchers. Its strength lies in providing a comprehensive suite of algorithms and functionalities, enabling deeper analytical customization and seamless integration with existing data science workflows. PM4Py's flexibility in handling diverse event log formats and its extensibility through Python's rich library ecosystem make it an invaluable tool for complex, data-driven process analysis.

A crucial distinction highlighted by this study concerns the performance of process discovery algorithms, specifically the Alpha Miner and Heuristic Miner, both implemented in PM4Py. While the Alpha Miner, a foundational algorithm, is theoretically sound, its limitations with real-world event logs—particularly those containing complex process patterns like short-loops, non-free-choice constructs, or duplicate tasks—were evident, resulting in a fitness of approximately 67%. In contrast, the Heuristic Miner demonstrated superior robustness and applicability in handling the inherent noise and complexity of real-world data. By employing heuristics to filter out infrequent behavior and focus on significant causal relationships, the Heuristic Miner achieved a remarkably higher fitness of 91%. This stark difference in performance unequivocally proves that the Heuristic Miner is more suitable for practical, real-world process discovery scenarios where event logs often contain imperfect or noisy data, delivering more accurate and interpretable process models compared to the more rigid Alpha Miner.

Ultimately, the choice between Disco and PM4Py, and by extension, the selection of underlying process discovery algorithms, hinges on the specific requirements and analytical objectives of a given project. For straightforward exploration and user-friendly visualization, Disco proves highly effective. However, for in-depth analysis, algorithmic flexibility, and handling the intricacies of real-world event logs, particularly as evidenced by the superior performance of heuristic approaches, PM4Py offers a more robust and scalable solution. Both tools, with their unique strengths, significantly contribute to advancing the field of process mining and empowering organizations to gain actionable insights from their operational data.

Bibliography

- [Aal11] Wil M. P. van der Aalst. “Process Mining: Discovery, Conformance and Enhancement of Business Processes”. In: *Springer Science & Business Media* (2011).
- [BZA19] Alessandro Berti, Sebastiaan J. van Zelst, and Wil M. P. van der Aalst. “Process Mining for Python (PM4Py): Bridging the Gap Between Process- and Data Science”. In: *arXiv preprint arXiv:1905.06169* (2019). URL: <https://arxiv.org/abs/1905.06169>.
- [Man17] Felix Mannhardt. *Hospital Billing - Event Log*. <https://doi.org/10.4121/uuid:76c46b83-c930-4798-a1c9-4be94dfeb741>. Version 1. 4TU.ResearchData. 2017.
- [WAW06] Annie M. M. Weijters, Wil M. P. van der Aalst, and Mathias Weske. “Process mining with the heuristic miner”. In: *IEEE Transactions on Knowledge and Data Engineering* 18.12 (2006), pp. 1629–1640.