Explorative Process Discovery Using Activity Projections

Yisong Zhang(✉) and Wil M. P. van der Aalst(✉)

Chair of Process and Data Science (PADS), RWTH Aachen University, Aachen, Germany
{zhang,wvdaalst}@pads.rwth-aachen.de

Abstract. This paper presents a tool to Explore Process Discovery (EPD) results using activity projection. Our EPD-Tool aims at exploring quality changes after removing activities from an event log. The main idea is to create a projected event log for every non-empty subset of activities and apply process discovery and conformance checking on them. The tool has been implemented as a plugin in ProM. First, EPD-Tool uses a process discovery algorithm to discover Petri net models for each projected event log. Then, EPD-Tool uses a conformance checking technique to compute conformance measures for each projected event log and model pair (L, N), e.g., fitness, precision, and $\mathcal{F}_1$-score. Finally, a dendrogram is generated to visualize the relationship between each log-model pair, thus enabling the systematic exploration of the different models using the dendrogram to find the best-performing node, i.e., a best log-model pair. This method prioritizes activities and detects redundancy in the process, which contributes to process enhancement. Conversely, critical activities are uncovered to help to shorten the processing time or save the process cost. This paper presents the EPD-Tool implementation and some example results.

Keywords: Process mining · Petri nets · Log projection · ProM

1 Introduction

After obtaining event logs from the underlying information systems, stakeholders can use *process mining* techniques to uncover their actual processes, provide insights, diagnose problems, and automatically trigger corrective actions [15,16]. *Process discovery* is a crucial step and the most challenging process mining task, since it aims to learn a process model from example behavior recorded in an event log. Each event in such a log refers to an activity, a well-defined step in some process, a process instance (case), and a timestamp. Process models discovered from event data show the actual process, e.g., the ordering of activities, frequencies, exceptional paths, and bottlenecks.

After obtaining a process model, we evaluate the quality of this model using several measures. Two widely-used control-flow-based *quality criteria* are replay *fitness* and *precision*. Fitness indicates how well the model reflects the behavior of the log. Precision reflects whether the model allows for additional unobserved behavior that is unlikely given the data.

State-of-the-art process discovery technologies [1–6] focus on the entire event log, and they ignore the impact of individual activity in the whole process. For instance, it is hard to prioritize activities in an event log using a traditional process discovery method, and the method of classifying which activities are redundant or critical is still missing. Therefore, we propose *Explorative Process Discovery using Activity Projections* (EPD) in this paper. This method uses activity-based projection to extract the sub-logs, and discovers the process model after deleting any activity in the event log. Afterward, a conformance checking technique records the changes in process model quality before and after deleting an activity. According to the comparison, we can prioritize activities and judge whether the activity is redundant or critical. We fully implemented the approach using the ProM [7] framework.

The remainder of the paper is structured as follows. In Sect. 2, we introduce basic concepts. Section 3 describes the approach. Section 4 presents the implementation of EPD-Tool and shows how to use this tool. Section 5 evaluates our approach using various data sets. Section 6 concludes the paper.

2 Preliminaries

In this section, we introduce some basic concepts and notations related to our research. The first and most important thing is the event log. Event logs serve as the starting point for any process mining task. An event log is a multiset of traces that describe the life cycle of cases in terms of the activities executed.

Definition 1 (Event Log). *Let $\mathcal{U}_{act}$ be the activity universe, i.e., the set of all possible activity attributes of events. A trace $\sigma = \langle a_1, a_2, \ldots, a_n\rangle \in \mathcal{U}_{act}^*$ is a sequence of activities. An event log $L \in \mathcal{B}(\mathcal{U}_{act}^*)$ is a multiset of traces. $\mathcal{U}_L = \mathcal{B}(\mathcal{U}_{act}^*)$ is the universe of event logs.*

$L_1 = [\langle a, b, c\rangle^5, \langle a, b, c, d, d\rangle^3, \langle a, d\rangle^2]$ and $L_2 = [\langle a, b, c, d\rangle^5, \langle a, b, c, d, d\rangle^3, \langle a, d\rangle^2]$ are two examples of an event log.

Definition 2 (Activity Projection). *Let $L \in \mathcal{U}_L$ be an event log and $\mathcal{A} \subseteq \mathcal{U}_{act}$ be a subset of activities. A projected event log is an event log where all activities not in $\mathcal{A}$ are removed. The projection function is $L{\upharpoonright_{\mathcal{A}}} = [\sigma{\upharpoonright_{\mathcal{A}}} \mid \sigma \in L]$ where $\sigma{\upharpoonright_{\mathcal{A}}}$ is defined recursively: (1) $\langle\rangle{\upharpoonright_{\mathcal{A}}} = \langle\rangle$ and (2) for $\sigma \in L$:*

$$(\langle a\rangle \cdot \sigma){\upharpoonright_{\mathcal{A}}} = \begin{cases} \sigma{\upharpoonright_{\mathcal{A}}} & \text{if } a \notin \mathcal{A} \\ \langle a\rangle \cdot \sigma{\upharpoonright_{\mathcal{A}}} & \text{if } a \in \mathcal{A} \end{cases} \tag{1}$$

where $\langle a\rangle \cdot \sigma$ appends activity a to trace σ.

Consider L_1 and L_2 introduced before, given $\mathcal{A}_1 = \{b, c\}$ and $\mathcal{A}_2 = \{a, d\}$, then $L_1{\upharpoonright_{\mathcal{A}_1}} = [\langle b, c\rangle^8, \langle\rangle^2]$ and $L_2{\upharpoonright_{\mathcal{A}_2}} = [\langle a, d\rangle^7, \langle a, d, d\rangle^3]$.

Process discovery techniques aim to learn a formal process model based on example behaviors in the event log [15,16]. There is a plethora of process modeling notations, e.g., Business Process Model and Notation (BPMN) [8], Process Trees, etc. Most of these modeling notations can be directly transformed into a Petri net [9], allowing for a

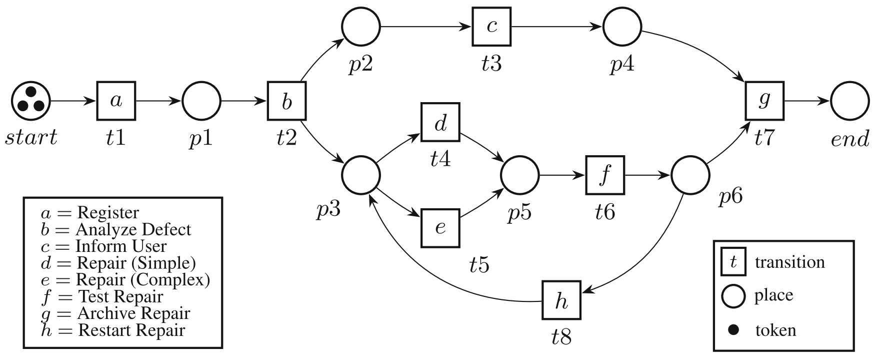


Fig. 1. Labeled Petri net N_1: A product repair process

range of analysis techniques including conformance checking and performance analysis. Therefore, we focus on *labeled accepting Petri nets*, i.e., Petri nets where transitions have activity labels and there is a well-defined initial and final marking. Note that we also allow for silent activities, i.e., transitions that do not have a label and that cannot be observed in the event log. This way, we can also model skipping and handle gateways in BPM and operators in process trees. $\mathcal{U}_N$ is the universe of labeled accepting Petri nets. An example of a labeled Petri net is depicted in Fig. 1.

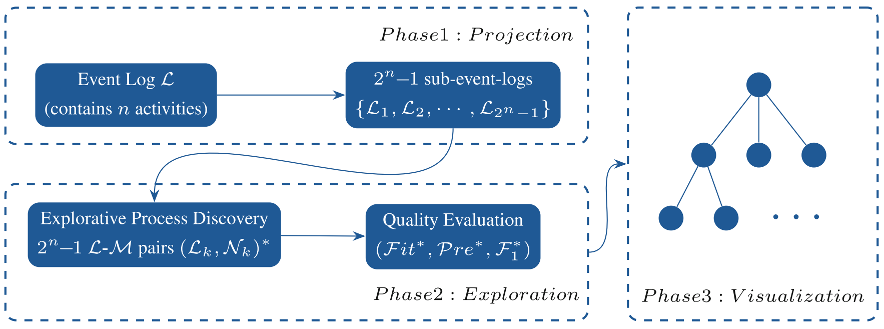
Our approach does not focus on any specific process discovery algorithm. Instead, *it can use any existing process discovery algorithm provided that it can be converted into a labeled accepting Petri net*. For instance, this paper uses Inductive Miner - infrequent (IMi) [10]. IMi takes an event log as input and discovers a process tree as output which could be transformed into a Petri net directly.

Definition 3 (Exploratory Process Discovery). *$disc : \mathcal{U}_L \rightarrow \mathcal{U}_N$ is a function that discovers a labeled accepting Petri net for any event log. Given an event log $L \in \mathcal{U}_L$, we can discover a model $N = disc(L)$. (L, N) is a log-model pair. Given a collection of event log we can create a collection of log-model pairs.*

For any log-model pair (L, N) conformance checking techniques are used to evaluate the quality of process models. In this paper, we use *alignments* [11] to compute replay *fitness* and *precision* [12] of each log-model pair. The $\mathcal{F}_1$-score is based on these.

Definition 4 (Quality Measures). *Let $(L, N) \in \mathcal{U}_L \times \mathcal{U}_N$ be a log-model pair. $\mathcal{F}_{it}(L, N) \in [0, 1]$ measures fitness (indicating how well the model reproduces the behavior of the log). $\mathcal{P}_{re}(L, N) \in [0, 1]$ measures precision (indicating to what degree the model's behavior is likely given the log). The harmonic mean of fitness and precision $\mathcal{F}_1(L, N) \in [0, 1]$ is defined as follows: $\mathcal{F}_1(L, N) = 2\frac{\mathcal{F}_{it}(L,N)\cdot\mathcal{P}_{re}(L,N)}{\mathcal{F}_{it}(L,N)+\mathcal{P}_{re}(L,N)}$.*

Here we abstract from the exact computation of fitness and precision and use the alignment-based fitness and precision values implemented in ProM [11,12].



$Phase1 : Projection$
Event Log $\mathcal{L}$ (contains n activities)
2^n–1 sub-event-logs $\{\mathcal{L}_1, \mathcal{L}_2, \cdots, \mathcal{L}_{2^n-1}\}$
Explorative Process Discovery 2^n–1 $\mathcal{L}$-$\mathcal{M}$ pairs $(\mathcal{L}_k, \mathcal{N}_k)^*$
Quality Evaluation $(\mathcal{F}it^*, \mathcal{P}re^*, \mathcal{F}_1^*)$
$Phase2 : Exploration$
$Phase3 : Visualization$

Fig. 2. Overview of explorative process discovery using activity projections.

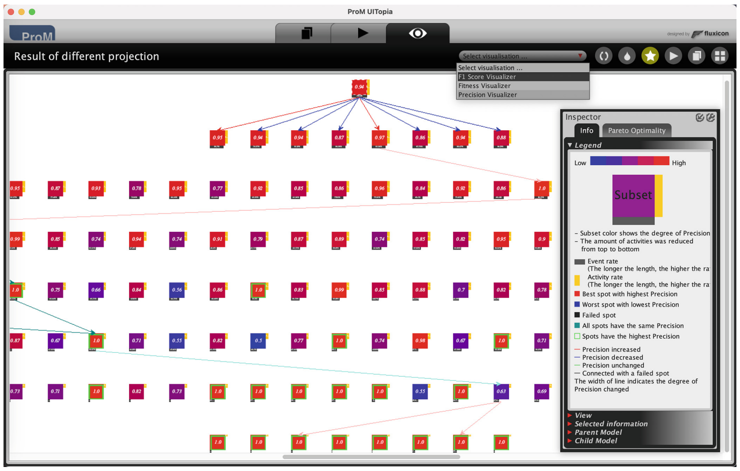
3 Approach

Since traditional process discovery methods focus on the entire event log and ignore the impact of each activity in the process, they cannot prioritize activities nor classify which activities are redundant or critical. Thus, we propose our "Explorative Process Discovery using Activity Projections" method to address this problem. The approach consists of the following three phases, as Fig. 2 shows:

- **Phase 1**: Projection. Given an event log $L \in \mathcal{U}_L$ containing n activities, for each subset of activities $\mathcal{A}_k$, where $\mathcal{A}_k \subseteq \mathcal{U}_{act}$ and $\mathcal{A}_k \neq \emptyset$, we use activity projection defined in Definition 2 to get a projected event log. Therefore, there are $2^n - 1$ sub-logs $\{L_1, L_2, \cdots, L_{2^n-1}\}$.
- **Phase 2**: Exploration. For each projected event log L_k, we use Explorative Process Discovery as in Definition 3 to discover a Petri net model $disc(L_k) = N_k$, there will be $2^n - 1$ log-model pairs (L_1, N_1), (L_2, N_2), $\cdots$, (L_{2^n-1}, N_{2^n-1}) where $(L_k, N_k) \in \mathcal{U}_L \times \mathcal{U}_N$. Then we use the quality measures described in Definition 4 to compute fitness, precision, and harmonic mean for each pair (L_k, N_k): $\mathcal{F}_{it}(L_k, N_k)$, $\mathcal{P}_{re}(L_k, N_k)$, and $\mathcal{F}_1(L_k, N_k)$.
- **Phase 3**: Visualization. Finally, we visualize the relationship between log-model pairs in a dendrogram and color it using the quality measures to explore the impact of removing a specific activity on the process model.

4 Implementation

The approach has been implemented as a plugin in the ProM framework, named "Explorative Process Discovery using Activity Projections" in the package "ExplorativeProcessDiscovery". To install EPD-Tool, simply download the latest ProM Nightly build from https://promtools.org/, run the PackageManager, select the package "ExplorativeProcessDiscovery", and run ProM. Now the user can import any event log data and apply the plugin "Explorative Process Discovery using Activity Projections". Our tool includes two versions of the function, "Full version" and "Lite version". The main difference between them is that the Full version handles all projected event logs simultaneously, and the Lite version requires users to configure step-by-step to explore



- Subset color shows the degree of Precision
- The amount of activities was reduced from top to bottom

Event rate (The longer the length, the higher the ra

Activity rate (The longer the length, the higher the ra

🟥 Best spot with highest Precision

🟦 Worst spot with lowest Precision

⬛ Failed spot

🟩 All spots have the same Precision

□ Spots have the highest Precision

— Precision increased

— Precision decreased

— Precision unchanged

— Connected with a failed spot

The width of line indicates the degree of Precision changed

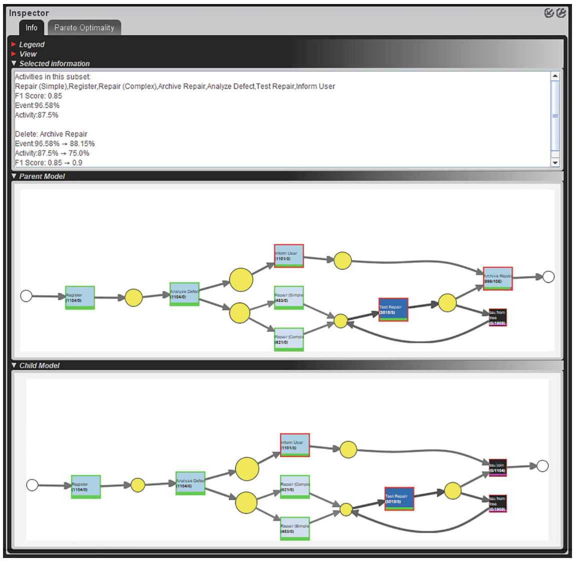
Fig. 3. The main interface of the Full version. Each node corresponds to a log-model pair. Only the connections that improve quality are shown.

the changes in model quality after removing any activity. Moreover, the Full version includes a Pareto optimal model of all the results, while the Lite version does not. We need these two versions because the Full version is time-consuming when processing large [13] event logs, while the Lite version improves efficiency by discarding some insignificant nodes.

4.1 Full Version Discovering Pareto Optimal Models

As Fig. 3 shows, we use a tree structure to show every projected event log simultaneously, and users can choose to color the dendrogram by fitness, precision, or $\mathcal{F}_1$-score through the drop-down list on the upper right. The best route for removal is always shown in this dendrogram. When a node is selected, it will display the connections to its child nodes. For more details, there is a floating "Inspect" window which also includes a "View" panel used to control the zoom function since the generated dendrograms are usually quite large.

- **Square box**: Each node corresponds to each projected event log.
  - **Color**: colors from blue to red; deep blue indicates low quality (fitness, precision, or harmonic mean), while the more red the color is, the better the quality is.
  - **Bottom grey rectangle**: the proportion of events left after projection; the longer the length is, the higher the proportion is.
  - **Right yellow rectangle**: the proportion of activities left after projection, the longer the length, the higher the proportion.
  - **Red box**: best node(s) with the highest quality.



Activities in this subset:
Repair (Simple),Register,Repair (Complex),Archive Repair,Analyze Defect,Test Repair,Inform User
F1 Score: 0.85
Event:96.58%
Activity:87.5%

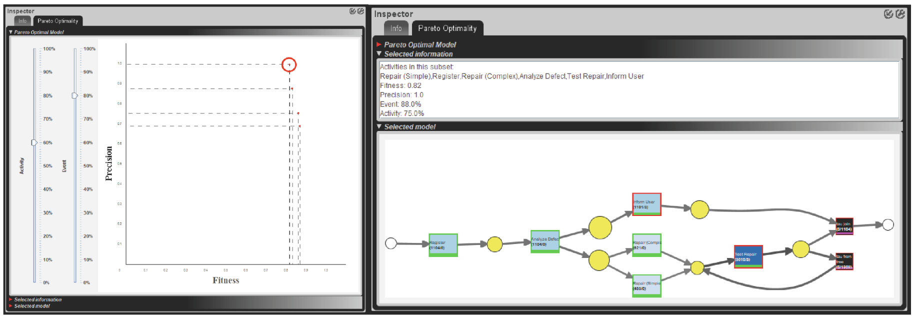
Delete: Archive Repair
Event:96.58% → 88.15%
Activity:87.5% → 75.0%
F1 Score: 0.85 → 0.9

Fig. 4. Detailed information is provided when selecting an edge connecting two models.

  - **Blue box**: worst node(s) with the lowest quality.
  - **Black box (optional)**: there may exist some log-model pairs that cannot proceed with alignments under limited resources (both RAM and time).
  - **Green box (optional)**: all nodes will be colored green if they all have the same best quality.
  - **Green border**: the best node(s) will be marked with a green border.
- **Line**: connect related nodes, the width indicates the degree of quality improvement/degradation after deleting the corresponding activity from the upper node to the lower node.
  - **Red line**: quality increased.
  - **Blue line**: quality decreased.
  - **Green line**: quality unchanged.
  - **Black line (optional)**: connected with a "Black box".

By selecting an edge between each pair of nodes, users can collect more information about this node pair, such as which activities are included in this subset, which element was deleted from the parent node, and the performance change between this node pair. Also, users can have an overview of the Petri net models of this node pair, as Fig. 4 shows. We aim to find a "sweet spot" among all projected event logs with the highest fitness and precision. However, such a "sweet spot" is hard to obtain in some cases. Therefore, the concept of Pareto optimality is used to guide the user.

Pareto Optimality. When multiple evaluation indicators exist, an object that is best on all evaluation indicators does not always exist. The concept of Pareto optimality aims to



Activities in this subset:
Repair (Simple),Register,Repair (Complex),Analyze Defect,Test Repair,Inform User
Fitness: 0.82
Precision: 1.0
Event: 88.0%
Activity: 75.0%

(a) (b)

Fig. 5. (a) The window of “Pareto optimality”. (b) Detailed information of selected “sweet spot”.

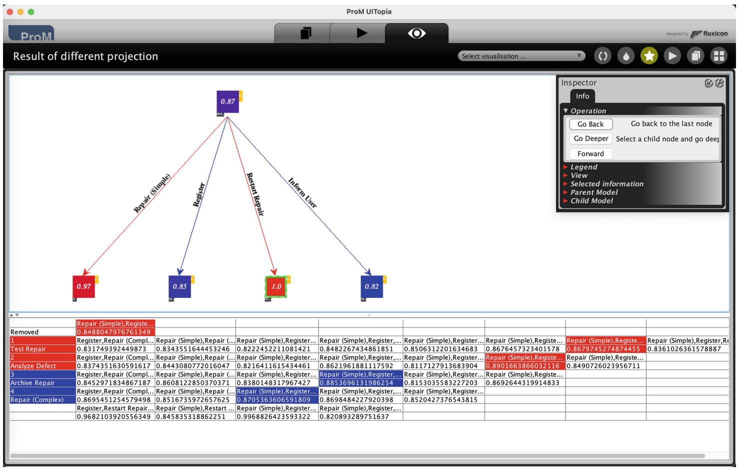
achieve a trade-off between those indicators, i.e., none of these indicators can be better without making at least one worse.

For most event logs, there is no node with both the highest fitness and the highest precision (except for nodes with only one activity). However, having a process with only one activity makes little sense. Therefore, as Fig. 5(a) shows, in the “Pareto Optimal Model”, the user can adjust the sliders to set the ratio of activities and events she wants to keep. After this, the tool will extract and show the “sweet spots” based on the concept of Pareto optimality, which contains the node with the highest fitness, the node with the highest precision, and a set of nodes with a trade-off between fitness and precision. Similarly, the specific information of each projected event log and the discovered Petri net model are visualized by selecting the corresponding node, as Fig. 5(b) shows.

4.2 Lite Version for Guided Exploration

As mentioned above, the Full version will be time-consuming when faced with large event logs, so we also provide a “Lite version” to improve the efficiency of the tool in some cases. As shown in Fig. 6, the dendrogram has only one layer of sub-nodes in this version. The user needs to configure to explore further sub-nodes step by step. Therefore, further operations are introduced in the inspector of the Lite version, such as “Go back”, “Go deeper”, and “Forward”.

- **Go back**: When the interface shows a deep layer, users can select “Go back” to return to the previous layer.
- **Go deeper**: Select “Go deeper” after choosing any child node to explore the deep layer of this child node. Note that it may take a while to display the results after selecting. Because the Lite version calculates each layer of nodes separately by selecting “Go deeper”.
- **Forward**: After selecting “Go back”, users have a chance to return to the deep layer without recalculation by selecting “Forward”. This improves efficiency in some cases because returning to the deep layer through “Go deeper” requires recalculation.



	Repair (Simple),Registe...							
Removed	0.8488047976761349							
1	Register,Repair (Compl...	Repair (Simple),Repair (...	Repair (Simple),Register...	Repair (Simple),Register,...	Repair (Simple),Register...	Repair (Simple),Registe...	Repair (Simple),Registe...	Repair (Simple),Register,Re
Test Repair	0.831749392449873	0.8343551644453246	0.8222452211081421	0.8482267434861851	0.8506312201634683	0.8676457323401578	0.8679745274874455	0.8361026361578887
2	Register,Repair (Compl...	Repair (Simple),Repair (...	Repair (Simple),Register...	Repair (Simple),Register,...	Repair (Simple),Register...	Repair (Simple),Registe...	Repair (Simple),Registe...	
Analyze Defect	0.8374351630591617	0.8443080772016047	0.8216411615434461	0.8621961881117592	0.8117127913683904	0.8901663866032116	0.8490726023956711	
3	Register,Repair (Compl...	Repair (Simple),Repair (...	Repair (Simple),Register...	Repair (Simple),Register,...	Repair (Simple),Register...	Repair (Simple),Registe...		
Archive Repair	0.8452971834867187	0.8608122850370371	0.8380148317967427	0.8853696131986254	0.8153035583227203	0.8692644319914833		
4	Register,Repair (Compl...	Repair (Simple),Repair (...	Repair (Simple),Register...	Repair (Simple),Register,...	Repair (Simple),Register...			
Repair (Complex)	0.8695451254579498	0.8516735972657625	0.8705363606591809	0.8698484227920398	0.8520427376543815			
	Register,Restart Repair...	Repair (Simple),Restart ...	Repair (Simple),Register...	Repair (Simple),Register,...				
	0.9682103920556349	0.845835318862251	0.9968826423593322	0.820893289751637				

Fig. 6. The main interface of the Lite version.

All other functions in the Lite version are the same as the Full version, including viewing the detailed information of sub-nodes and checking the Petri net model. The only difference is that the Lite version can record the best removals in a table, and the exploration of Pareto optimality models is not supported.

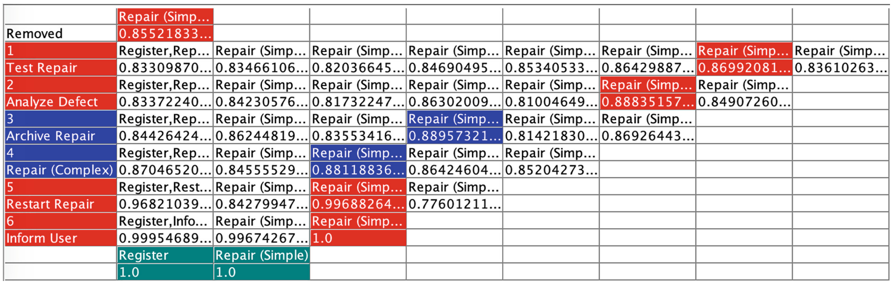
5 Evaluation

In this section, we conduct experiments using two data sets "Repair[1]" and "Road Traffic Fine Management Process" [14] to evaluate our approach. It includes two parts: (1) evaluations of the general functions of our tool and (2) an explanation of why we need a "Lite version" by comparing the time required by the two versions.

5.1 General Functions

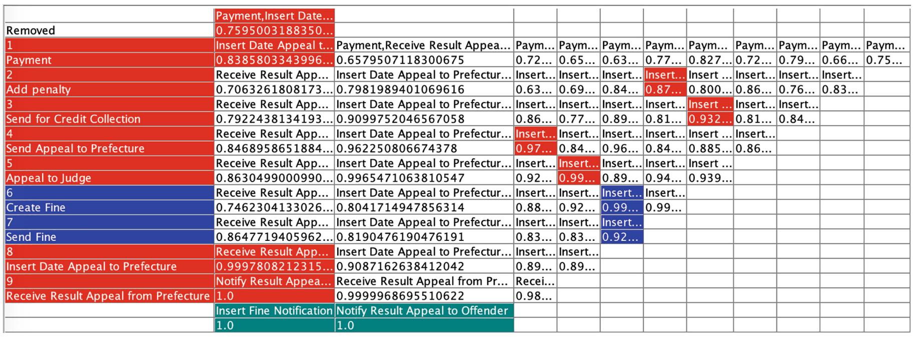
First, for a general function of our tool, we can define an activity route or the priority of activities. As shown in Fig. 7 and Fig. 8, we use $\mathcal{F}_1$-score as the evaluation criteria. Here we call the best-performing sub-log the "sweet spot". Each row from top to bottom of these tables records the performance change after removing an activity from the previous sweet spot. Colored cells indicate the sweet spot for the corresponding layer, red means performance increased, and blue means performance decreased. Column "Removed" indicates the activities removed from the previous sweet spot to get the current sweet spot. As a result, the priority of activities in "Repair" is {(Register, Inform User), Restart Repair, Repair (Complex), Repair (Simple), Archive Repair, Analyze Defect, Test Repair}, which means the most frequent (stable) activity in "Repair" is "Register" or "Inform User", and if organizations want to reduce the process or detect

[1] https://processmining.org/old-version/files/repairexample.zip.



	Repair (Simp...							
Removed	0.85521833...							
1	Register,Rep...	Repair (Simp...	Repair (Simp...	Repair (Simp...	Repair (Simp...	Repair (Simp...	Repair (Simp...	Repair (Simp...
Test Repair	0.83309870...	0.83466106...	0.82036645...	0.84690495...	0.85340533...	0.86429887...	0.86992081...	0.83610263...
2	Register,Rep...	Repair (Simp...	Repair (Simp...	Repair (Simp...	Repair (Simp...	Repair (Simp...	Repair (Simp...	
Analyze Defect	0.83372240...	0.84230576...	0.81732247...	0.86302009...	0.81004649...	0.88835157...	0.84907260...	
3	Register,Rep...	Repair (Simp...	Repair (Simp...	Repair (Simp...	Repair (Simp...	Repair (Simp...		
Archive Repair	0.84426424...	0.86244819...	0.83553416...	0.88957321...	0.81421830...	0.86926443...		
4	Register,Rep...	Repair (Simp...	Repair (Simp...	Repair (Simp...	Repair (Simp...			
Repair (Complex)	0.87046520...	0.84555529...	0.88118836...	0.86424604...	0.85204273...			
5	Register,Rest...	Repair (Simp...	Repair (Simp...	Repair (Simp...				
Restart Repair	0.96821039...	0.84279947...	0.99688264...	0.77601211...				
6	Register,Info...	Repair (Simp...	Repair (Simp...					
Inform User	0.99954689...	0.99674267...	1.0					
	Register	Repair (Simple)						
	1.0	1.0						

Fig. 7. The best route to remove problematic activities for the data set "Repair".



	Payment,Insert Date...										
Removed	0.7595003188350...										
1	Insert Date Appeal t...	Payment,Receive Result Appea...	Paym...	Paym...	Paym...	Paym...	Paym...	Paym...	Paym...	Paym...	Paym...
Payment	0.8385803343996...	0.6579507118300675	0.72...	0.65...	0.63...	0.77...	0.827...	0.72...	0.79...	0.66...	0.75...
2	Receive Result App...	Insert Date Appeal to Prefectur...	Insert...	Insert...	Insert...	Insert...	Insert ...	Insert...	Insert...	Insert...	
Add penalty	0.7063261808173...	0.7981989401069616	0.63...	0.69...	0.84...	0.87...	0.800...	0.86...	0.76...	0.83...	
3	Receive Result App...	Insert Date Appeal to Prefectur...	Insert...	Insert...	Insert...	Insert...	Insert ...	Insert...	Insert...		
Send for Credit Collection	0.7922438134193...	0.9099752046567058	0.86...	0.77...	0.89...	0.81...	0.932...	0.81...	0.84...		
4	Receive Result App...	Insert Date Appeal to Prefectur...	Insert...	Insert...	Insert...	Insert...	Insert ...	Insert...			
Send Appeal to Prefecture	0.8468958651884...	0.962250806674378	0.97...	0.84...	0.96...	0.84...	0.885...	0.86...			
5	Receive Result App...	Insert Date Appeal to Prefectur...	Insert...	Insert...	Insert...	Insert...	Insert ...				
Appeal to Judge	0.8630499000990...	0.9965471063810547	0.92...	0.99...	0.89...	0.94...	0.939...				
6	Receive Result App...	Insert Date Appeal to Prefectur...	Insert...	Insert...	Insert...	Insert...					
Create Fine	0.7462304133026...	0.8041714947856314	0.88...	0.92...	0.99...	0.99...					
7	Receive Result App...	Insert Date Appeal to Prefectur...	Insert...	Insert...	Insert...						
Send Fine	0.8647719405962...	0.8190476190476191	0.83...	0.83...	0.92...						
8	Receive Result App...	Insert Date Appeal to Prefectur...	Insert...	Insert...							
Insert Date Appeal to Prefecture	0.9997808212315...	0.9087162638412042	0.89...	0.89...							
9	Notify Result Appea...	Receive Result Appeal from Pr...	Recei...								
Receive Result Appeal from Prefecture	1.0	0.9999968695510622	0.98...								
	Insert Fine Notification	Notify Result Appeal to Offender									
	1.0	1.0									

Fig. 8. The best remove route for the data set "Road Traffic Fine Management Process".

problems of this process, they should start from "Test Repair". More specifically, the infrequent (unstable) activities might be {Test Repair, Analyze Defect} because the performance increased after removing these activities, and the performance will decrease if we keep removing any other activities.

Similarly, the priority of activities in "Road Traffic Fine Management Process" is {(Notify Result Appeal to Offender, Insert Fine Notification), Receive Result Appeal from Prefecture, Insert Date Appeal to Prefecture, Send Fine, Create Fine, Appeal to Judge, Send Appeal to Prefecture, Send for Credit Collection, Add penalty, Payment}, that means the most frequent (stable) activity in "Road Traffic Fine Management Process" is "Notify Result Appeal to Offender" or "Insert Fine Notification". If organizations want to reduce the process or detect problems in this process, they should start with "Payment", and the infrequent (unstable) activities might be {Payment, Add penalty, Send for Credit Collection, Send Appeal to Prefecture, Appeal to Judge}.

Moreover, to describe the function of "Pareto optimality" more intuitively, we use the data set "Repair" as a demonstration. As shown in Fig. 5(a), in Pareto optimality, we set the activity and event thresholds to 60% and 80%, respectively, and extracted 4 "sweet spots". Consider the "sweet spot" marked with a red circle, as detailed in Fig. 5(b). After removing the activity Archive Repair, the model's $\mathcal{F}_1$-score changes from 0.85 to 0.9. This result indicates that "Archive Repair" might be an infrequent

Table 1. Comparison of time cost between two versions.

	Data size		Time cost	
	Events	Activities	Full version	Lite version
Repair	11,855	8	38.9s	7.2s
RTFMP	561,470	11	**11,724.9**s	161.0s

(unstable) activity in this process. Therefore, organizations can optimize the whole process by focusing on this activity.

5.2 Scalability

To explain the necessity of the "Lite version" more clearly, we compare the time required by the full and Lite versions to get the same result of priority and classification. As Table 1 shows, for the Repair log, the processing time in the Lite version is around one-tenth that in the Full version, and for RTFMP (Road Traffic Fine Management Process), it decreased from more than 3 h to less than 3 min. (Experiment conducted using an 11th Gen Intel Core i7-1165G7 2.8GHz processor and 16GB RAM.) Although the Lite version does not display all the results at once, it saves much time to observe the impact of removing a specific activity. Additionally, users can still explore the priority of activities step by step instead of just waiting.

6 Conclusion

This paper introduced a new tool for process discovery named the Explorative Process Discovery tool using Activity Projections (EPD-Tool), which is implemented as a ProM plugin. With this plugin, users can explore the impact of removing any activity from the event log on the model. EPD-Tool provides users with insight to identify redundant or critical activities. Therefore, they can optimize processes and reduce process costs based on their expert knowledge. In the future, we plan to refine this tool to improve efficiency and integrate more features to provide users with a deeper insight into the event log incorporating performance related to time and resources (to find problematic resources and time consuming activities). Moreover, there are ways to further improve the scalability of the tool.

Acknowledgments. The authors thank the Alexander von Humboldt (AvH) Stiftung for supporting this research. Funded by the Deutsche Forschungsgemeinschaft (DFG) under Germany's Excellence Strategy, Internet of Production (390621612).

References

1. Augusto, A., Conforti, R., Dumas, M., et al.: Automated discovery of process models from event logs: review and benchmark. IEEE Trans. Knowl. Data Eng. **31**(4), 686–705 (2018)
2. De Weerdt, J., Vanden Broucke, S., Vanthienen, J., et al.: Active trace clustering for improved process discovery. IEEE Trans. Knowl. Data Eng. **25**(12), 2708–2720 (2013)
3. Goedertier, S., Martens, D., Vanthienen, J., et al.: Robust process discovery with artificial negative events. J. Mach. Learn. Res. **10**, 1305–1340 (2009)
4. Vanden Broucke, S., De Weerdt, J.: Fodina: a robust and flexible heuristic process discovery technique. Decis. Support Syst. **100**, 109–118 (2017)
5. Ghose, A., Koliadis, G., Chueng, A.: Process discovery from model and text artifacts. IEEE Congress on Services (Services 2007), 167–174. IEEE (2007)
6. Slaats, T.: Declarative and hybrid process discovery: recent advances and open challenges. J. Data Semant. **9**(1), 3–20 (2020)
7. van Dongen, B.F., de Medeiros, A.K.A., Verbeek, H.M.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: The ProM framework: a new era in process mining tool support. In: Ciardo, G., Darondeau, P. (eds.) ICATPN 2005. LNCS, vol. 3536, pp. 444–454. Springer, Heidelberg (2005). https://doi.org/10.1007/11494744_25
8. Chinosi, M., Trombetta, A.: BPMN: an introduction to the standard. Comput. Stand. Interfaces **34**(1), 124–134 (2012)
9. Murata, T.: Petri nets: properties, analysis and applications. Proc. IEEE **77**(4), 541–580 (1989)
10. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from incomplete event logs. In: Ciardo, G., Kindler, E. (eds.) PETRI NETS 2014. LNCS, vol. 8489, pp. 91–110. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07734-5_6
11. Van der Aalst, W.M.P., Adriansyah, A., Van Dongen, B.F.: Replaying history on process models for conformance checking and performance analysis. Wiley Interdisc. Rev.: Data Min. Knowl. Discov. **2**(2), 182–192 (2012)
12. Adriansyah, A., Munoz-Gama, J., Carmona, J., van Dongen, B.F., van der Aalst, W.M.P.: Alignment based precision checking. In: La Rosa, M., Soffer, P. (eds.) BPM 2012. LNBIP, vol. 132, pp. 137–149. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36285-9_15
13. Leemans, S.J.J., Fahland, D., Van der Aalst, W.M.P.: Scalable process discovery and conformance checking. Softw. Syst. Mod. **17**(2), 599–631 (2018)
14. de Leoni, M. (Massimiliano); Mannhardt, Felix (2015): Road Traffic Fine Management Process. 4TU.ResearchData Dataset. https://doi.org/10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5
15. Van der Aalst, W.M.P.: Process Mining: Data Science in Action. Springer-Verlag, Berlin (2016). https://doi.org/10.1007/978-3-662-49851-4
16. Van der Aalst, W.M.P., Carmona, J. (eds.): Lecture Notes in Business Information Processing, vol. 448. Springer-Verlag, Berlin (2022)