ProReco: A Process Discovery Recommender System

Tsung-Hao Huang[1][0000−0002−3011−9999], Tarek Junied[2][0000−0001−9318−9276],
Marco Pegoraro[1][0000−0002−8997−7517], and
Wil M. P. van der Aalst[1][0000−0002−0955−6940]

[1] Process and Data Science (PADS), RWTH Aachen University, Aachen, Germany
{tsunghao.huang,pegoraro,wvdaalst}@pads.rwth-aachen.de
http://www.pads.rwth-aachen.de/
[2] RWTH Aachen University, Aachen, Germany
tarekjunied@icloud.com

Abstract. Process discovery aims to automatically derive process models from historical execution data (event logs). While various process discovery algorithms have been proposed in the last 25 years, there is no consensus on a dominating discovery algorithm. Selecting the most suitable discovery algorithm remains a challenge due to competing quality measures and diverse user requirements. Manually selecting the most suitable process discovery algorithm from a range of options for a given event log is a time-consuming and error-prone task. This paper introduces `ProReco`, a **Pro**cess discovery **Reco**mmender system designed to recommend the most appropriate algorithm based on user preferences and event log characteristics. ProReco incorporates state-of-the-art discovery algorithms, extends the feature pools from previous work, and utilizes eXplainable AI (XAI) techniques to provide explanations for its recommendations.

Keywords: Process Mining · Process Discovery · Recommender System · Explainable Recommendations · Explainable AI.

1 Introduction

Process discovery [1] is a discipline that aims to automatically obtain formal representation through models of the operating mechanisms in a process. The input of such methods is a collection of data related to the historical execution of a process, often in the form of discrete *events*. Discovery algorithms read events and their *attributes* from a dataset (often called an *event log*), and output a process model, to provide a representation as close as possible to the real process operations.

Since the inception of the discipline in the early 2000s, many discovery algorithms have been proposed [2], as well as numerous metrics to assess their desirability and quality. Nevertheless, the systematic review and benchmark [3] show no algorithm dominating all other methods in terms of model quality measures. Moreover, producing a satisfactory process model is still an open challenge, although there exists extensive literature dedicated to measuring the quality of models obtained through discovery. This is because (i) some of the most widely adopted quality measures are competing (i.e., there exist trade-offs between them), and (ii) depending on the final use of the discovered model, different (and sometimes opposite) characteristics are desirable. Under such a circumstance, users are left with the task of manually selecting the most prominent process discovery algorithm for the event log at hand. The procedure is time-consuming and error-prone even for process mining experts, let alone inexperienced users.

To address the aforementioned problems and to assist process mining users, previous works [6,7] proposed using recommender systems for process discovery. The approaches [6,7] abstract from the actual values of model quality by calculating the final score based on the rankings. Also, the approach in [7] does not incorporate user preferences for the recommendation, assuming every user wants to maximize all measures simultaneously. Lastly, the recommendations offered by both works lack accompanying explanations. Intransparent recommendations could hamper the acceptance of a recommender system [9].

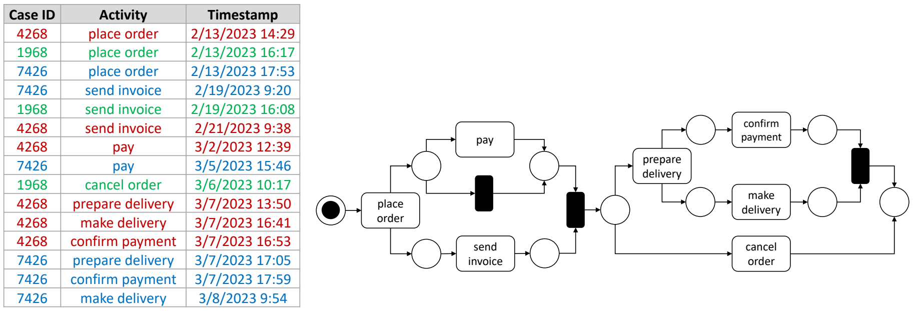
This paper proposes `ProReco`, a **Pro**cess discovery **Reco**mmender system. Given an event log and user preferences regarding model quality measures, `ProReco` recommends the most appropriate process discovery algorithm tailored to the users' needs. Internally, `ProReco` utilizes machine learning models to predict the values for each quality measure before computing the weighted (user preferences) sum of the final score. The scores are then used to rank and recommend the discovery algorithm. `ProReco` not only expands the features pool from previous work but also includes state-of-the-art process discovery algorithms. Last but not least, for every recommendation made by `ProReco`, explanations are available to the user thanks to the incorporation of the eXplainable AI (XAI) technique [5] in `ProReco`.

The remainder of the paper is structured as follows. Section 2 illustrates some preliminary notions. Section 3 describes the components and mechanics of `ProReco`. Lastly, Section 4 concludes the paper and indicates directions for future research.

2 Preliminary Concepts

In this section, we introduce the necessary concepts before presenting `ProReco` in Sec. 3.

Event log The starting point of process mining is the event log where each event refers to a case (an instance of the process), an activity, and a point in time. The existence of these three attributes is the minimal requirement for an event log, whereas more attributes can be recorded and/or extracted. Event data can be extracted from various sources such as a database, a transaction log, a business suite/ERP system, etc. An event log can be seen as a collection of cases, whereas a case is a trace/sequence of events. Fig. 1a shows a synthetic event log for the purchasing process of an online retail site. Each row corresponds to an event.



Case ID	Activity	Timestamp
4268	place order	2/13/2023 14:29
1968	place order	2/13/2023 16:17
7426	place order	2/13/2023 17:53
7426	send invoice	2/19/2023 9:20
1968	send invoice	2/19/2023 16:08
4268	send invoice	2/21/2023 9:38
4268	pay	3/2/2023 12:39
7426	pay	3/5/2023 15:46
1968	cancel order	3/6/2023 10:17
4268	prepare delivery	3/7/2023 13:50
4268	make delivery	3/7/2023 16:41
4268	confirm payment	3/7/2023 16:53
7426	prepare delivery	3/7/2023 17:05
7426	confirm payment	3/7/2023 17:59
7426	make delivery	3/8/2023 9:54

(a) An example event log

(b) A process model represented using Petri net

Fig. 1: An example of an event log and the corresponding process model.

Process model A process model is a structured representation of the activities and their relationships within a business process. It plays a crucial role in understanding, analyzing, and improving organizational workflows. Various process modeling notations exist such as Petri nets, BPMNs, BPEL models, or UML Activity Diagrams [1]. In `ProReco`, we focus on Petri net since it is one of the simplest formalisms that explicitly model concurrency. Moreover, it is trivial to convert process models in other notations into Petri nets.

Fig. 1b shows the corresponding process model (in the form of a Petri net) for the event log in Fig. 1a. The process starts with the activity *"place order"* followed by the concurrent executions of activity *"pay"* and *"send invoice"*, where activity *"pay"* is optional. Then, the process might be either *"cancel"* or followed by a delivery procedure.

Process discovery Process discovery aims at constructing process models to describe the observed behaviors of information systems from event logs. In general, the problem of process discovery can be defined as follows: A process discovery algorithm is a function that maps an event log L onto a process model N such that the model N is representative of the behaviors seen in the log L. Despite the development of process discovery algorithms, manually finding the most appropriate algorithm is a challenging and error-prone task. To assist users with identifying the most prominent discovery algorithm, we present `ProReco` in the next section.

3 ProReco: A Process Discovery Recommender System

The backend of `ProReco` is developed in Python, to leverage the capabilities of the PM4py[3] package. The package provides a comprehensive suite of algorithms and tools for process mining. The source code for `ProReco` can be found on a GitHub repository[4], which provides detailed instruction for installation. In

[3] https://pm4py.fit.fraunhofer.de/

[4] https://github.com/TarekJunied/ProReco

the following, we introduce the structure and the main functions of ProReco. Additionally, a demo video of ProReco is available[5].

3.1 Structure

The overall structure of ProReco is shown in Fig. 2. To recommend the most prominent discovery algorithm for event log L, ProReco takes a vector of weights W representing the importance of different measures in addition to L. The weights (within the interval [0,100]) are given by the users and will be used to calculate the final score of the algorithm.

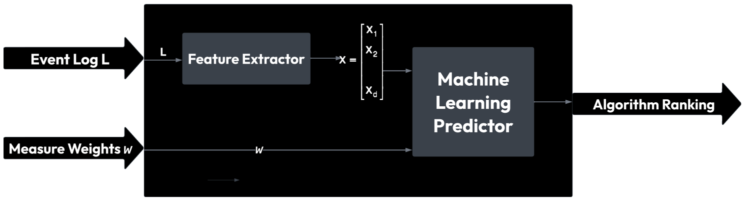


Fig. 2: General structure of ProReco

The output of ProReco is a ranking for each discovery algorithm in our portfolio, as well as the corresponding score calculated based on the weighted sum of the quality measures. The higher the score, the better the algorithm adapts to the users' preferences. In the following, we briefly describe the target quality dimensions used in ProReco.

As we aim to quantify the most common quality measures of a process model, the four primary quality measures [1] (*fitness*, *simplicity*, *precision*, and *generalization*) are used. A model with good fitness represents (and can replay) behavior seen in the log. The simplicity dimension refers to the complexity of the model. In the context of process mining, this means that a simpler model is advantageous, as long as it can explain the behaviors seen in the log. A precise process model does not allow too much unseen behavior, as it is trivial to create a model that allows any behavior (the flower model [1]). Lastly, a model with good generalization can represent behavior unseen in the event log. Since the four quality dimensions compete with each other, a single ideal model often does not exist [1]. The ideal model highly depends on the use case of the users. This motivates the use of weights to incorporate the user preferences w.r.t. the importance of the four quality measures.

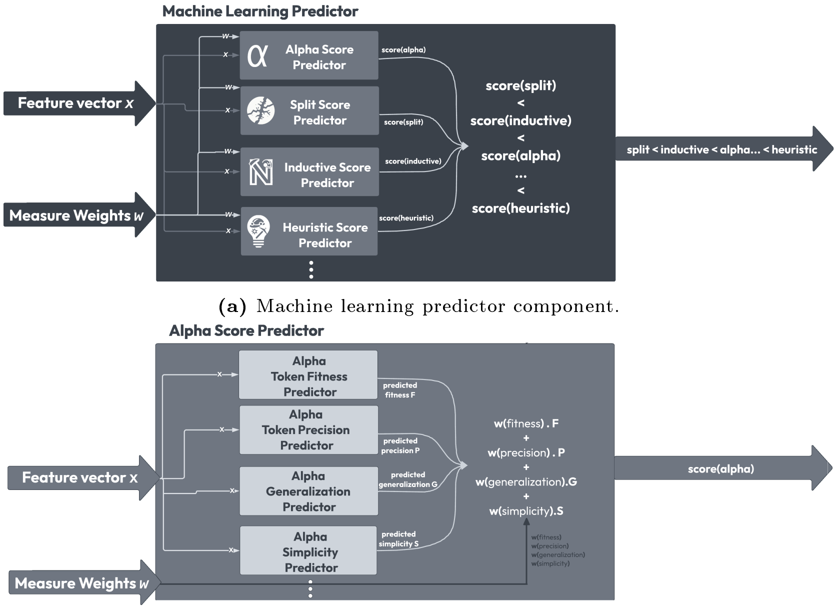
Next, we introduce each component (the feature extractor and the machine learning predictor) in more detail.

Feature Extractor Based on previous work [6,7,8], we extract an initial pool of various features. Moreover, the initial pool is filtered considering two criteria. First, we remove the computationally expensive features. As efficiency is one

[5] https://bit.ly/prorecodemo

of the motivations for developing such a recommender system, using features that are expensive to compute does counteract the benefit. Second, we remove redundant features. Features are considered redundant if there is already another feature representing the same concept. For instance, the feature representing the number of trace variants is implemented as `n_unique_traces` in [7] and as "Number of distinct traces" in [6]. These redundancies lead to higher execution time and adversely affect the performance of some machine learning models. Thus, we remove such redundancies using the Pearson correlation coefficient. Lastly, we add ten Directly-Follows Graph (DFG)- and footprint matrix-based features. In the end, 162 features were extracted. The introduction to all features is out of scope. The corresponding function (called *Featurer*) providing insight for all features is available in `ProReco` and introduced in Sec. 3.2 in more detail.

Machine Learning Predictor As shown in Fig. 3a, the machine learning predictor consists of a score predictor for each algorithm in the algorithm portfolio, which consists of Alpha Miner, Alpha-Plus Miner, Heuristics Miner, Inductive Miner (classic, infrequent, direct), ILP Miner, and Split Miner.


Machine Learning Predictor
Feature vector x
Measure Weights w
w
x
Alpha Score Predictor
Split Score Predictor
Inductive Score Predictor
Heuristic Score Predictor
score(alpha)
score(split)
score(inductive)
score(heuristic)
score(split) < score(inductive) < score(alpha) ... < score(heuristic)
split < inductive < alpha... < heuristic
Alpha Score Predictor
Alpha Token Fitness Predictor
Alpha Token Precision Predictor
Alpha Generalization Predictor
Alpha Simplicity Predictor
predicted fitness F
predicted precision P
predicted generalization G
predicted simplicity S
w(fitness) . F + w(precision) . P + w(generalization).G + w(simplicity).S
w(fitness)
w(precision)
w(generalization)
w(simplicity)
score(alpha)

(a) Machine learning predictor component.

(b) Score predictor for a single algorithm (Alpha Miner as an example)

Fig. 3: Machine learning predictor and its sub-components: score predictors.

The structure for an algorithm score predictor is shown in Fig. 3b using Alpha Miner as an instance. The Score Predictor consists of individual predictors for each of the four measures (fitness, precision, generalization, and simplicity). Each measure predictor accepts a feature vector derived from the event log as input and forecasts the value of the corresponding measure for the process model that would have been generated based on the provided event log. The measure weights W provided by the user are then used for the final computation of the overall score for a discovery algorithm. During this computation, each predicted measure value is multiplied by its corresponding measure weight and subsequently aggregated.

The choice of the predictor to predict the measure values for each algorithm is of little importance here, as it is flexible to switch among different predictors whenever suitable. In ProReco, we adopt the xgboost [4] regressor as an instantiation for the predictor. To train the predictors, we included 12 real-life event logs from the 4TU repository[6] and 785 synthetic event logs generated by the PLG tool[7]. The data is available for download[8]. We used 5-fold cross-validation for each experiment with an 80/20 training/test split.

3.2 ProReco's Functions

In this section, we introduce the main functions of ProReco.

Recommendation As a recommender system for process discovery, ProReco recommends the most prominent algorithm for the user according to the predicted weighted sum of the four quality measures discussed in Sec. 3.1. The inputs are an event log L and the user preferences w.r.t. measure weights W.

To initiate the recommendation, users have to upload an event log (.xes format) as input. Then, they are redirected to a page where they have to provide the weights for each of the four quality measures. Once the measure weights are submitted, users are redirected to the recommendation page, where a ranking of the algorithm portfolio, the score for each algorithm, and the predicted measure values for each algorithm and measure are available. Additionally, ProReco provides users the possibility to mine a process model with the recommended process discovery algorithms. The discovered process model is then displayed through an interactive Petri net viewer.

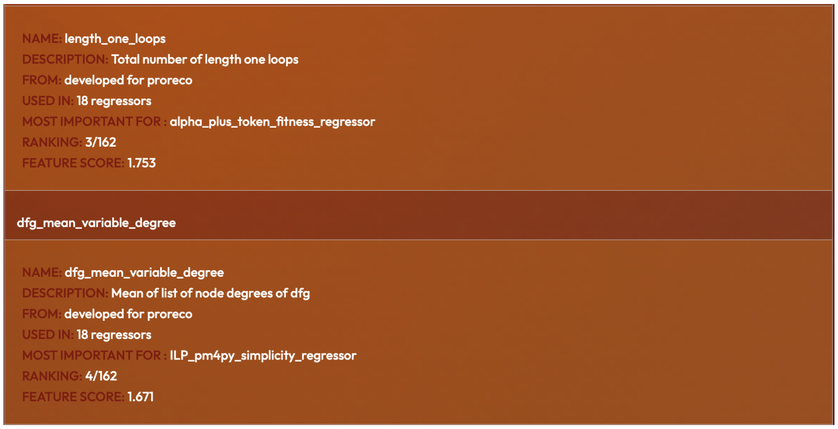
Feature Insights ProReco offers insights into the features extracted from event logs. The *"Featurer"* (shown in Fig. 4) is accessible through the navigation bar. *Featurer* provides the user with detailed information about the features. By searching for a specific feature name, users can access the following information, as shown in Fig. 4:

- Description: a brief description of the feature.
- From: the source of the feature.
- Used in: the number of regressors that use this feature.
- Most important for: the regressor that gains the most advantage from the feature.
- Ranking: the importance of the feature among all features.
- Feature Score: a metric used to determine the feature's ranking.

[6] https://data.4tu.nl/

[7] https://plg.processmining.it/

[8] http://bit.ly/allEventLogsProReco

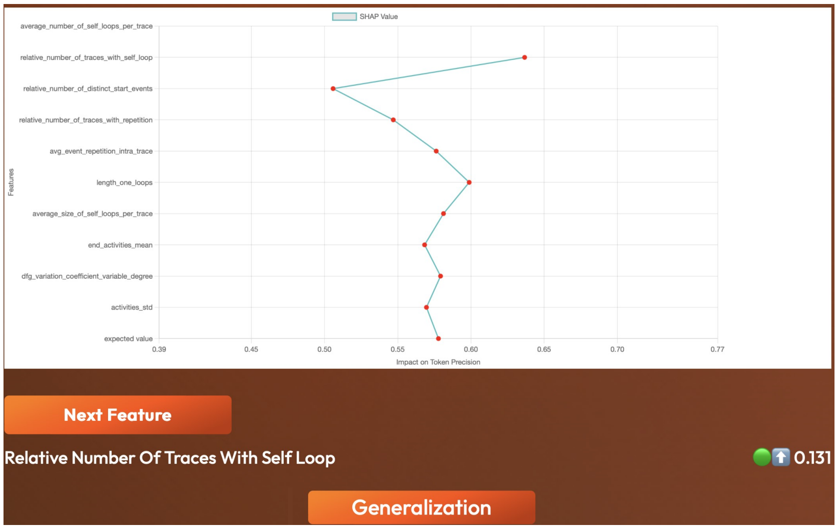


NAME: length_one_loops
DESCRIPTION: Total number of length one loops
FROM: developed for proreco
USED IN: 18 regressors
MOST IMPORTANT FOR : alpha_plus_token_fitness_regressor
RANKING: 3/162
FEATURE SCORE: 1.753

dfg_mean_variable_degree

NAME: dfg_mean_variable_degree
DESCRIPTION: Mean of list of node degrees of dfg
FROM: developed for proreco
USED IN: 18 regressors
MOST IMPORTANT FOR : ILP_pm4py_simplicity_regressor
RANKING: 4/162
FEATURE SCORE: 1.671

Fig. 4: The user interface for the feature insights.


SHAP Value
average_number_of_self_loops_per_trace
relative_number_of_traces_with_self_loop
relative_number_of_distinct_start_events
relative_number_of_traces_with_repetition
avg_event_repetition_intra_trace
length_one_loops
average_size_of_self_loops_per_trace
end_activities_mean
dfg_variation_coefficient_variable_degree
activities_std
expected value
Features
0.39
0.45
0.50
0.55
0.60
0.65
0.70
0.77
Impact on Token Precision
Next Feature
Generalization

Relative Number Of Traces With Self Loop ● ⬆ 0.131

Fig. 5: The user interface for explaining an individual recommendation.

Explainable Recommendation As recommendations without explanations can hinder the transparency, trustworthiness, and satisfaction of a recommender system [9], `ProReco` incorporates techniques from explainable AI (XAI) to provide explanations for individual predictions made by the regressors. Users can access the explanations by clicking on "Explain the Predictions" on the resulting recommendation page. Then, users will be redirected to an interactive plot based on SHAP values [5]. The plot begins at the bottom, displaying the expected measure for the selected algorithm. As each feature is added, its effect on the prediction is shown. A shift to the left indicates a decrease in the measure, while a shift to the right indicates an increase. The interactive plot offers insights and explanations for the recommendations.

4 Conclusion and Future Work

Process discovery aims to automatically generate process models representing the underlying information system from event logs. Despite the development of various discovery algorithms and quality metrics, no single algorithm dominates in terms of model quality measures [3]. Consequently, users often face the challenge of manually selecting suitable discovery algorithms, a process that is time-consuming and error-prone, even for experts in process mining. In response, this paper introduces `ProReco`, a process discovery recommender system designed to recommend the most appropriate process discovery algorithm based on user preferences and event log characteristics. ProReco expands upon previous work by incorporating state-of-the-art algorithms and providing transparent explanations for its recommendations.

As future work, several directions can be investigated. First, we plan to explore various parameter settings for the discovery algorithms. Due to the vast search space, the challenge is finding an efficient method to determine the best value per technique. Additionally, we would like to include additional measures for optimization such as runtime. Last but not least, we plan to conduct user studies to understand the effectiveness and usability of `ProReco` as well as to validate the benefits. For example, the understandability of the provided explanation could be evaluated by a dedicated user study.

Acknowledgements. We thank the Alexander von Humboldt (AvH) Stiftung for supporting our research.

References

1. van der Aalst, W.M.P.: Process Mining - Data Science in Action, Second Edition. Springer (2016)
2. Augusto, A., Carmona, J., Verbeek, E.: Advanced process discovery techniques. In: van der Aalst, W.M.P., Carmona, J. (eds.) Process Mining Handbook, Lecture Notes in Business Information Processing, vol. 448, pp. 76–107. Springer (2022)
3. Augusto, A., Conforti, R., Dumas, M., Rosa, M.L., Maggi, F.M., Marrella, A., Mecella, M., Soo, A.: Automated discovery of process models from event logs: Review and benchmark. IEEE Trans. Knowl. Data Eng. **31**(4), 686–705 (2019)
4. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: KDD. pp. 785–794. ACM (2016)
5. Lundberg, S.M., Lee, S.: A unified approach to interpreting model predictions. In: NIPS. pp. 4765–4774 (2017)
6. Ribeiro, J., Carmona, J., Misir, M., Sebag, M.: A recommender system for process discovery. In: BPM. Lecture Notes in Computer Science, vol. 8659, pp. 67–83. Springer (2014)
7. Tavares, G.M., Junior, S.B., Damiani, E.: Automating process discovery through meta-learning. In: CoopIS. Lecture Notes in Computer Science, vol. 13591, pp. 205–222. Springer (2022)
8. Zandkarimi, F., Decker, P., Rehse, J.R.: Fig4pm: A library for calculating event log measures (extended abstract) (2021)
9. Zhang, Y., Chen, X.: Explainable recommendation: A survey and new perspectives. Found. Trends Inf. Retr. **14**(1), 1–101 (2020)