A Recommender System for Process Discovery

Joel Ribeiro, Josep Carmona, Mustafa Misir, Michèle Sebag

A Recommender System for Process Discovery

Joel Ribeiro[1], Josep Carmona[1], Mustafa Mısır[2], and Michele Sebag[2]

[1] Universitat Politècnica de Catalunya, Spain.
{jribeiro, jcarmona}@lsi.upc.edu
[2] TAO, INRIA Saclay - CNRS - LRI, Universite Paris Sud XI, Orsay, France.
{mustafa.misir,michele.sebag}@lri.fr

Abstract. Over the last decade, several algorithms for process discovery and process conformance have been proposed. Still, it is well-accepted that there is no dominant algorithm in any of these two disciplines, and then it is often difficult to apply them successfully. Most of these algorithms need a close-to expert knowledge in order to be applied satisfactorily. In this paper, we present a recommender system that uses portfolio-based algorithm selection strategies to face the following problems: to find the best discovery algorithm for the data at hand, and to allow bridging the gap between general users and process mining algorithms. Experiments performed with the developed tool witness the usefulness of the approach for a variety of instances.

Keywords: Process Mining, Recommender Systems, Algorithm Selection

1 Introduction

The ability of monitoring process executions within information systems yields large-scale event log files. These files can be processed using the so-called *process mining* approaches, at the crossroad of *business intelligence* and *data mining* techniques. Process mining is positioning as the perfect candidate to support information systems in the *big data* era.

Process mining is defined as the extraction of valuable information from event logs, aimed at strategic insight into the business processes [13]. Process mining mainly includes *process discovery*, *conformance checking* and *enhancement*. Discovery techniques aim at the behavioral modeling of the business process underlying the event logs. Conformance techniques check the compatibility of a process model with regard to a set of event logs. Enhancement techniques enrich a process model based on additional process information available in the event log.

This paper focuses on process discovery, acknowledged to be the most challenging issue in process mining. While several algorithms have been proposed for process discovery (e.g., the reader can find a complete summary in [13]), there is no algorithm dominating all other algorithms. Furthermore, these algorithms are built on different formalisms (e.g., Petri nets, BPMN, EPC, Causal nets).

The selection of the process discovery algorithm and formalism most appropriate to (a set of) event logs is left to the user, hindering the deployment of the process mining approach in two ways. On the one hand, inexperienced users can

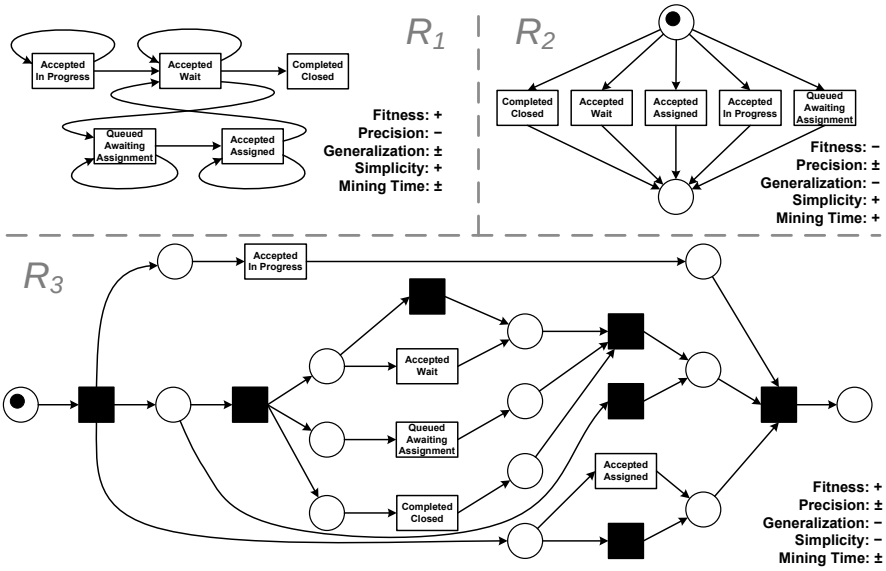


Fig. 1: The process discovery problem: three discovered models for a given log. R_1 is a Causal net discovered by the Flexible Heuristic Miner (FHM), while R_2 and R_3 are Petri nets, discovered by the Alpha and Inductive miners, respectively. These control-flow algorithms are available in the ProM 6 framework [16].

hardly get the best of an algorithm portfolio. On the other hand, experienced users might have to manually inspect the event log to select the appropriate algorithm, along a tedious, time-consuming and error-prone procedure. Figure 1 illustrates the problem: three different models were discovered by three different techniques using the same log. Each model is annotated with a set of generic quality measurements (+: good, ±: average, −: poor; for an overview of quality measures see Section 3.3). Depending on the measurements in consideration, one model may be preferred with respect to the others. If all measurements were considered, the technique presented in this paper would recommend model R_1 (i.e., recommend the FHM). However, if only fitness and precision were considered, model R_3 (Inductive miner) would be recommended by our technique.

The contribution of the paper is an integrated process discovery framework achieving algorithm selection based on machine learning techniques. Formally, this framework elaborates on the *Algorithm Recommender System* (ARS) approach [5], based on using a dataset that reports the results of some algorithms in the portfolio on a set of problem instances; its generality is witnessed as it has been applied successfully in domains such as *Constraint Satisfaction* and *Optimization*.

ARS is integrated within a framework to evaluate process discovery algorithms [14]. We have developed a server-client architecture along the training-test principle used in machine learning. The server achieves lifelong learning, continuously running process discovery experiments to enrich its database reporting the performances of algorithms on case studies (event logs). This database is exploited

using ARS, continuously increasing the system knowledge. This knowledge is then disseminated to the clients, that use it to predict the best algorithm on their current event log. The client is implemented as a ProM [16] plugin (Nightly Build version)[3], named `RS4PD` under the `Recommendation` package. Experiments using real-life and artificial logs confirm the merits of the proposed approach.

The remainder of this paper is organized as follows. Section 2 presents background and discusses related work. Section 3 presents an overview of the recommender system; its implementation is detailed in Section 4. Section 5 provides an experimental validation of the approach. Section 6 contains a preliminary study about the selection of parameters for discovery algorithms, while Section 7 concludes the paper with some discussion and perspectives for further research.

2 Related Work

Over the last decade, recommender systems became present in a significant number of applications of information systems [2]. In spite of this, few attempts have been done on recommending process mining algorithms. In this paper, the main goal is to build a system for recommending process discovery algorithms. The proposed recommender system requires the combination of three different disciplines. We overview them now in the following subsections.

2.1 Evaluation of Process Discovery Algorithms

Control-flow discovery algorithms focus on finding the causality of activities within a process, e.g., order, conflict, concurrency, iteration, among others. Several approaches can be found in the literature [13]. These algorithms (or the resulting models) can be evaluated using conformance techniques [10], which may reveal mismatches between observed and modeled behavior. Rozinat et al. [9] identified the need of developing a methodology to benchmark process mining algorithms. A conceptual framework was then proposed to evaluate and compare process models. Weber et al. [18] proposed a methodology for assessing the performance of process mining algorithms. This approach assumes the generation of event logs from reference models for applying conformance analysis. Also assuming the existence of reference models, Wang et al. [17] proposed a framework for evaluating process mining algorithms based on the behavioral and structural similarities between reference and mined process models. In this approach, the information gathered from the evaluation (i.e., the similarities between process models) is then used to support a recommender system for process mining algorithms. A different evaluation approach for analyzing the quality of process models was introduced by vanden Broucke et al. [14]. In this approach, several conformance checking metrics can be computed over an event log and a process model in an integrated environment.

[3] `http://www.promtools.org/prom6/nightly/`

2.2 Collaborative Filtering

Due to a large number of choices regarding an item, it is hard to determine possible personal choices without checking the available options throughly. Recommender systems are automated methods to efficiently perform this task. One way to do it is by using item or user related content data given beforehand. This sub-field of recommender systems is studied as content-based filtering methods. Instead of directly using such data, it is possible to employ users' earlier preferences on items. In this way, finding users with similar taste or items with similar user preferences is practical to make user-item predictions. *Collaborative filtering* is the field approaching the recommendation problems from this perspective [12]. The underlying motivation is that if some users share similar preference characteristics on a set of commonly known items, it is likely that these users will have similar taste on other items.

Algorithm Recommender System (ARS) [5] is an algorithm portfolio selection tool that uses collaborative filtering. ARS takes the user-item matrix idea into an instance-algorithm matrix indicating the performance of each algorithm on each instance. The *Algorithm Selection Problem* [8] has been targeted in different areas such as *Constraint Satisfaction* and *Optimization*. The methods developed on these contexts using algorithm selection, like SATZilla [20] and CPHydra [7], need a full performance dataset showing how well a set of algorithms performed on a set of problem instances. Besides that some of these methods were designed in a way that they can only be used for the problems with a specific performance criterion, such as runtime. Unlike these existing methods, ARS does not require a full performance matrix, thanks to collaborative filtering. In addition, ARS has a generic structure that can be used as a black-box method, thus it can be used for any algorithm selection task as the one we have in this paper. This is provided by using a rank-based input scheme. In particular, the performance database involves relative performance, i.e., ranks of tested algorithms on each instance.

2.3 Information Retrieval

Information Retrieval (IR) is a discipline that considers finding information that best satisfies some given criteria (e.g., keywords) on documents. Among the many techniques available, *top-k queries* is a technique used in the framework proposed in this paper. These queries can be defined as the search of the k most relevant (or interesting) entries in a multidimensional dataset. The first algorithms for efficient top-k queries are the so-called *threshold* algorithms [4]. Considered the reference algorithms in the subdomain, threshold algorithms rely on sequential and random accesses to information to compute the exact top-k results. Using an index-based approach to access information, Akbarinia et al. proposed two algorithms [1] that exploit the position of the entries in the dataset to compute the exact top-k results. From these algorithms, the BPA2 algorithm is used in this study for retrieving the top-k discovery techniques, due to its efficiency.

3 Overall Framework

A recommender system for process discovery can follow the same strategy as the portfolio-based algorithm selection [20]. Basically, this selection relies on a set (portfolio) of algorithms, which are executed over a repository of input objects (e.g., datasets or problems). Information about the executions (e.g., performance or results) is used to identify the best algorithms with regard to specific input objects. By characterizing these objects as sets of features, it is possible to build a *prediction model* that associates a ranking of algorithms with features. So, the prediction of the best-performing algorithms on a given input object can be achieved by first extracting the features of that object and then using the prediction model to compute the ranking of algorithms. This approach can be used to build a recommender system for process discovery, with event logs as input objects and discovery techniques as algorithms.

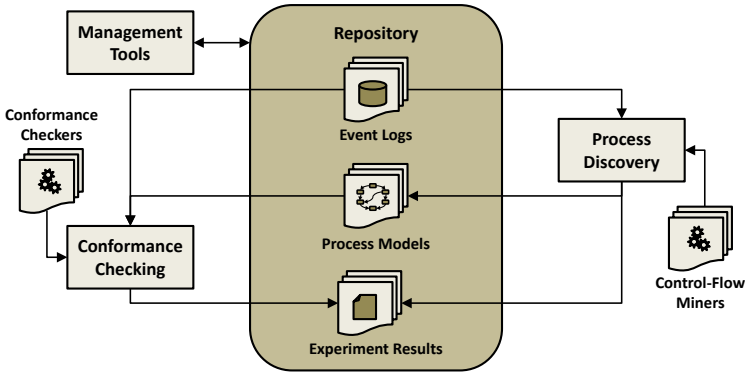


Fig. 2: Outline of the evaluation framework.

Figure 2 presents a framework for evaluating process discovery techniques, which can be used to support a recommender system. The *Process Discovery* and the *Conformance Checking* nodes represent the execution of a process mining experiment. These experiments can be defined as follows.

Discovery experiment: consists of executing a control-flow algorithm on an event log in order to produce a process model. The mined model as well as information about the algorithm performance are stored in the repository.

Conformance experiment: consists of computing a conformance measurement on a process model and the event log used to mine that model. The experiments results are stored in the repository.

The *Management Tools* allow (i) the execution of discovery and conformance experiments and (ii) the management of the repository as well as the collection of discovery and conformance techniques (i.e., the control-flow miners and the conformance checkers). The execution of an experiment is selected randomly. Applying this strategy, the insertion of event logs, control-flow miners, and conformance checkers can be done at any moment.

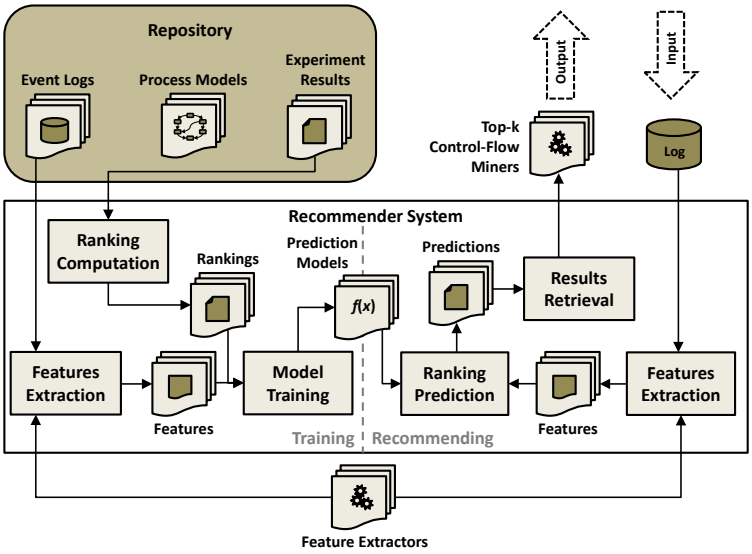


Fig. 3: Overview of the recommender system.

Figure 3 presents an overview of our recommender system. As depicted, the recommender system includes two functionalities: *training* and *recommending*. The training function generates the necessary knowledge from the experiment results to build prediction models. This can be achieved as follows.

i. The experiment results are retrieved from the repository.
ii. For each event log and measurement (performance or conformance) in the results, the ranking of discovery techniques is computed. A ranking of techniques must contain all control-flow miners used in the experiments. In the case a ranking is incomplete (i.e., there is not enough experiment results to compute a complete ranking), a machine learning algorithm (e.g., SVM or Neural Networks) is applied to predict the missing ranking values [5].
iii. The features of the log are extracted for each event log in the results.
iv. For each measurement in the results, the corresponding prediction model is trained using the rankings of discovery techniques and the features of the logs.

The recommending function uses the prediction models to obtain the top-k best-performing discovery techniques for an event log. This can be achieved as follows.

a. The features of the given event log are extracted.
b. For each prediction model, the ranking of techniques with respect to a measurement is predicted using the extracted features.
c. All the predicted rankings are combined into a final ranking.
d. The top-k techniques are retrieved from the final ranking.

The following sections describe in detail the key elements used in the training and recommending parts of the proposed system.

3.1 Features

A *feature* is a numerical function defined on event logs. A set of features therefore induces a (semi-)distance on event logs. In practice, a feature can be defined as a specific characteristic of the event log. By characterizing two logs as two sets of features, it is possible to assess whether or not the logs are different with regard to those features. This means that the execution of discovery techniques and the corresponding results can be associated to features of logs. Importantly, these associations can be used to identify which techniques perform better over logs characterized by specific features. A feature can be defined under one of three different scopes: *trace*, *event*, and *flow*.

Trace features: focus on characteristics of sequences of events. The average trace length is an example of a trace feature.

Event features: focus on characteristics of single events. The number of distinct events in the log is an example of an event feature.

Flow features: focus on characteristics of causal dependencies (i.e., pairs of consecutive events in the same trace). The number of one-length loops in the log is an example of a flow feature.

A challenge for building a recommender system for process discovery is the definition or selection of a representative set of features, supporting the algorithm selection. A representative set of features is described in Section 4; the validation, extension and improvement of the feature set is left for further study.

3.2 Techniques

A (discovery) *technique* consists of a control-flow algorithm for process discovery.[4] As in the portfolio-based algorithm selection, a set of techniques can be executed over a repository of event logs. The information gathered from the execution can be used to analyze which techniques perform best with regard to the performance of discovery techniques and the quality of their results. Remark that different techniques may produce different types of process models (e.g., the ILP miner produces a Petri net, while the FHM mines a Causal net). Since the conformance checking algorithms used in this study work only on Petri nets, a model conversion may be necessary in order to enable the results of a technique to be evaluated.

3.3 Measures

A *measure* can be defined as a measurement that evaluates the performance of discovery techniques and the quality of their results. By evaluating the execution of two discovery techniques over the same log (as well as the produced results), it is possible to identify which technique performs better with regard to some

[4] Remark that other process discovery perspectives such as the *resource*, the *time*, and the *data* perspectives are not considered in the present work. The integration of these perspectives in the recommender system is identified as future work.

measures. The recommender system proposed in this paper considers either a particular measure (aiming at identifying the best algorithm with regard to this measure), or an aggregation of these measures using an information retrieval algorithm (cf. Section 3.4). Together with the characteristics of the logs (i.e., the sets of features), this information can be used to build prediction models for supporting a recommendation system. A measure can be categorized as follows [13].

Performance measure: quantifies a discovery algorithm in terms of execution on a specific event log. The runtime is an example of a performance measure.

Simplicity measure: quantifies the results of a discovery algorithm (i.e., the process model mined from a specific event log) in terms of readability and comprehension. The number of elements in the model is an example of a simplicity measure.

Fitness measure: quantifies how much behavior described in the log complies with the behavior represented in the process model. The fitness is 100% if the model can describe every trace in the log.

Precision measure: quantifies how much behavior represented in the process model is described in the log. The precision is 100% if the log contains every possible trace represented in the model.

Generalization measure: quantifies the degree of abstraction beyond observed behavior, i.e., a general model will accept not only traces in the log, but some others that generalize these.

3.4 Recommending the Top-k Best-Performing Techniques

The recommendation of the top-k best-performing techniques for a specific event log is based on a set of ranking predictions. A ranking prediction identifies the techniques that are expected to perform better with regard to a specific measure. This information is computed using prediction models (i.e., functions that map a set of features to a ranking of techniques), which are built using the results of discovery and conformance experiments. The top-k best-performing techniques are then determined by a final ranking in which one or more ranking predictions are taken into account. The selection of the top-k techniques from the final ranking can be seen as a typical information retrieval problem.

4 Implementation

The implementation of the recommender system proposed in this paper is based on a server-client architecture. The main function of the server is to generate knowledge about the performance of techniques on different event logs. The server includes also both the evaluation framework and the repository, which support the training function of the recommender system. The training function as well as the evaluation framework are implemented as a package in the CoBeFra framework [15], while the repository is supported by a transactional database. The main function of the client is based on the knowledge generated in the server, and

consists of predicting (recommending) the best-performing techniques for a given event log. This function is implemented as a ProM plugin (available in ProM 6).

As depicted in Figure 2, the evaluation framework relies on a collection of discovery and conformance algorithms. The current portfolio consists of 9 discovery techniques, which can be evaluated using 8 conformance checking algorithms. Table 1 presents the initial collection of techniques of the recommender system. The conformance checking algorithms are used to assess the quality of the results of the techniques (i.e., the measures as defined in Section 3.3). Table 2 presents the initial set of measures that can be assessed in the recommender system. Remark that performance measures are generated in the discovery experiments, while all the other measures are computed in conformance experiments.

Technique	*Result*
Alpha Miner	Petri Net
Flexible Heuristics Miner	Causal Net
Flower Miner	Petri Net
Fuzzy Miner	Fuzzy Model
Heuristics Miner	Causal Net
Inductive Miner	Petri Net
ILP Miner	Petri Net
Passage Miner	Petri Net
TS Miner	Transition System

Table 1: Portfolio of control-flow algorithms. These algorithms are available in the ProM 6 framework [16].

Category	*Measure*
Performance	Runtime
	Used Memory
Simplicity	Elements in the Model
	Node Arc Degree
	Cut Vertices
Fitness	Token-Based Fitness
	Negative Event Recall
Precision	ETC Precision
	Negative Event Precision
Generalization	Neg. Event Generalization

Table 2: Set of measures. The conformance checking algorithms that support these measures are available in CoBeFra [15].

As depicted in Figure 3, both training and recommending functions rely on a set of feature extractors. A feature extractor consists of a relatively simple function that can be used to compute specific features of event logs. An initial collection of 12 extractors was implemented and integrated in the system. Table 3 describes the set of features that can be computed using these extractors. Remarkably, experiments presented in Section 5 suggest that, although simple, these features are very effective in the characterization of event logs.

To enable flexibility and extensionality, any technique, measure, or feature can be added to (or removed from) the system at any moment, even when some experiment is being executed. The modification (addition or removal of techniques, measures, or features) will have effect in the succeeding iteration of the training.

4.1 Evaluation Framework

The evaluation framework is implemented as a package of the CoBeFra framework and supported by a MySQL database management system (DBMS). The different functionalities of the framework can be described as follows.

Scope	*Feature*	*Description*
Trace	Distinct Traces	The number of distinct traces in the log.
Trace	Total Traces	The number of traces in the log.
Trace	Trace Length	The average length of all traces in the log.
Trace	Repetitions Intra Trace	The average number of event repetitions intra trace.
Event	Distinct Events	The number of distinct events in the log.
Event	Total Events	The number of events in the log.
Event	Start Events	The number of distinct start events in the log.
Event	End Events	The number of distinct end events in the log.
Flow	Entropy	The average of the proportion of direct successors and predecessors counts between two events in the log.
Flow	Concurrency	Based on the dependency measures of [19], the percentage of concurrent relations in the causal matrix.
Flow	Density	The percentage of non-zero values in the causal matrix.
Flow	Length-One Loops	The number of length-one loops in the causal matrix.

Table 3: The set of features. The causal matrix consists of the counting of direct successors for each pair of events in the log.

Management function: controls the repository as well as the collection of discovery and conformance algorithms. The repository consists of a database storing information about event logs, process models, and experiments. The discovery and conformance algorithms consist of executables (e.g., ProM plugins) that can be used for process discovery or conformance checking.

Execution function: executes a single evaluation by selecting randomly an event log, a control-flow algorithm (i.e., a technique), and a conformance algorithm from the repository and the collection of algorithms. An evaluation starts with either executing a discovery experiment in order to mine a process model using the selected discovery technique on the selected log or, if this discovery experiment was executed in a previous evaluation, retrieving this process model from the database. The execution of a discovery experiment consists of running the selected control-flow algorithm on the selected log in which a process model and the performance measures (cf. Table 2) are computed. Both mined model and measures are stored in the database.[5] The evaluation then continues with the execution of the conformance experiment (if possible), which consists of running the selected conformance algorithm on the selected log and mined model. As a result, a measure is computed and stored in the database.

4.2 Recommender System

Training The system's training function is implemented as a Java application. Invoked by a trigger (e.g., every Friday), this application retrieves all the infor-

[5] Only Petri net models are stored in the repository. If the result of a discovery experiment is not a Petri net then a conversion is necessary. For some model formalisms such as Causal nets, this can be achieved by invoking some ProM plugins. For other formalisms like Fuzzy models, no model will stored in the repository. This means that only performance measures can be computed for these cases.

mation about the experiments by querying the database. Then, the set of event logs referred in the query results is retrieved from the repository. For each log in the set, it is extracted the set of features (cf. Table 3) that characterizes the log. Next, the entries of the query results are grouped by measure. For each measure and log, a list of experiments results is created, ordered by the result value (e.g., the runtime).[6] This list is then used to build a ranking of techniques. A matrix containing the rankings of the measure is finally built. Each column of the matrix represents a technique, while each row refers to the log from which the ranking was computed. Next, the matrix completion of the ARS algorithm [5] is applied on the matrix to predict eventual unknown values. The matrix as well as the sets of features of the logs described in the matrix can then be used to build a model for predicting the ranking of techniques from a set of features.

Recommending The system's recommending function is implemented as a ProM plugin (cf. Figure 4). Invoked in the ProM framework, this plugin takes an event log as input and produces a *recommendation* of the best-performing discovery techniques for the given log. The recommendation is based on the knowledge produced by the system's training function. First, the given log is characterized as a set of features. Then, using these features and for each measure, it is applied the prediction function of the ARS algorithm [5] on the matrices generated in the training. As a result, a list of predicted rankings is returned, where each entry represents the expected best-performing techniques for a specific measure. The recommendation is based on a final ranking combining all the predicted rankings. The combined score of a technique $t \in T$ is defined by

$$score(t) = \sum_{m \in M} w_m \times rank(t, m),$$

where $m \in M$ is a measure, w_m is the weight of m, and $rank(t, m)$ is the position of t in the predicted ranking of m. Giving a list of prediction rankings and the weights of each measure, the top-k entries of the final ranking can be efficiently computed by applying the BPA2 algorithm [1].

5 Experiments

A set of experiments was conducted in order to evaluate the recommender system proposed in this paper. Using the implemented evaluation framework and recommender system, we first executed a number of experiments over a set of event logs in order to build the system's prediction models. For these experiments, 130 event logs (112 synthetic and 18 real life) were collected from several sources[7] and uploaded into the repository. As described in Section 4, the portfolio consisted

[6] For performance and simplicity measures, the list follows an ascending order. For the other measures, the list follows a descending order. One-element lists are discarded once that they do not hold enough comparative information.

[7] Several process mining groups were invited to share their event logs.

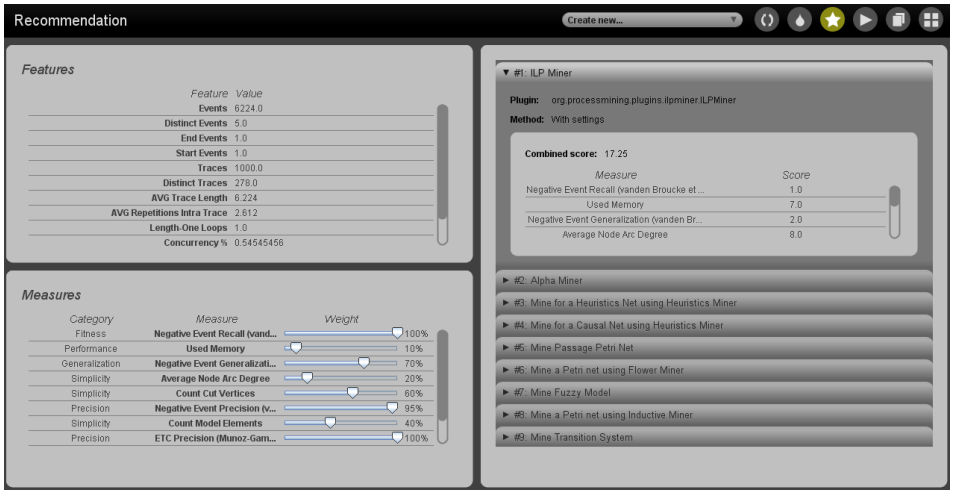


Fig. 4: `RS4PD`: the client as a ProM plugin. Top-left panel shows the features computed for the uploaded event log. Bottom-left panel allows the user to provide weights to each one of measures. Right panel shows the recommendation.

of 9 discovery techniques (cf. Table 1), which can be evaluated using 8 conformance checking algorithms. These conformance algorithms were used to compute the non-performance measures of Table 2. Remark that the performance measures are computed during the execution of the discovery experiments. The set of feature extractors used in the experiments is described in Table 3. The system's evaluation started with the continuous execution of experiments during one week. As a result, 1129 discovery experiments were executed, from which 882 process models were generated. In total, 5475 measures were computed.

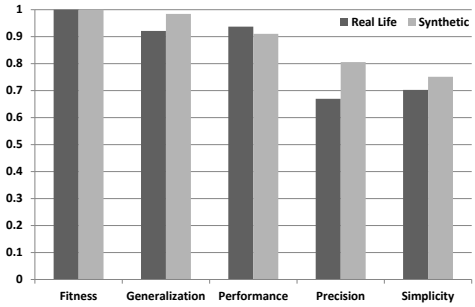
Using the prediction models built from the experiments, we then used a set of testing event logs in order to compare the accuracy of the system's recommendations. The testing dataset consists of 13 event logs from the 3TU repository[8] and the testing dataset of [19]. From these logs, 4 are the real life logs used in the *Business Process Intelligence* (BPI) workshop challenges of 2012 and 2013. For each of the testing event logs, we executed all the possible discovery and conformance experiments. Then, using the system's recommending function, we computed the top-9 best-performing techniques for each measure. The *accuracy* of the recommendation is defined by the matching of the predicted technique with the actual best-performing technique measured in the experiments.[9] The accuracy is 1 if the predicted best-performing technique matches the measured best-performing technique. The accuracy is 0 if the predicted best-performing technique matches the measured worst-performing technique. An accuracy value between 0 and 1 is defined by the min-max normalization of the measure value of the predicted best-

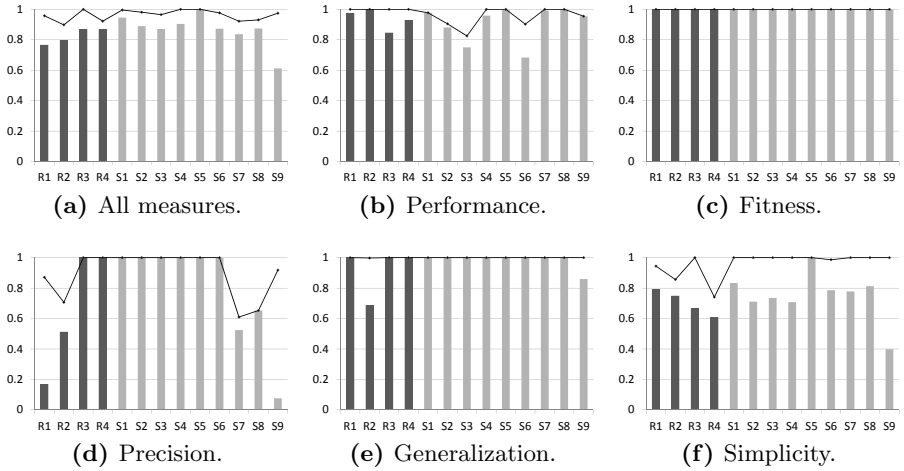
[8] `http://data.3tu.nl/repository/collection:event_logs`

[9] Remark that, unlike rank correlations such as Spearman's or Kendall's, this accuracy measurement does not consider the worst-performing techniques in the rankings, which are unlikely to be taken into account by the user.

performing technique, where min and max are the values of the measured worst- and best-performing techniques. The results of the assessment of the system's accuracy are shown in the figures bellow.

The figure on the right presents the average accuracy of the prediction of the best-performing technique for each measure category, discriminated by event log type. These results show that the system's accuracy varies from 0.67 (for precision measures on real life logs) to 1.0 (for fitness measures). Considering both all measures and all event log types, the global system's accuracy is 0.854.





(a) All measures. **(b)** Performance. **(c)** Fitness.

(d) Precision. **(e)** Generalization. **(f)** Simplicity.

Fig. 5: Average accuracy of the system's recommendation for each event log.

Figure 5 presents the average accuracy of the system's recommendation for each event log. The average accuracy of the prediction of the best-performing technique (i.e., the top-1 technique) is represented by the bars; these accuracy values are discriminated by event log type (dark gray bars represent real life logs, while synthetic logs are identified by the light gray bars). The average accuracy of the prediction of the best-performing technique taking into account the top-3 techniques is represented by the lines. The accuracy values for these cases are defined by the best matching between these three techniques and the actual best-performing technique measured in the experiments (i.e., one of the top-3 techniques should be the actual best-performing technique). Figure 5a shows the global system's accuracy, while the remaining figures show the system's accuracy for each measure category. These results show that for some logs the recommendation of a specific measure may not be accurate (e.g., precision measures on logs $R1$ and $S9$). Nevertheless, the global system's accuracy varies from 0.612 and 1.0.

Taking into account the top-3 techniques instead of the top-1, the lower bound of this accuracy interval increases to 0.898. Considering both all measures and all logs, the global system's accuracy considering the top-3 techniques is 0.963.

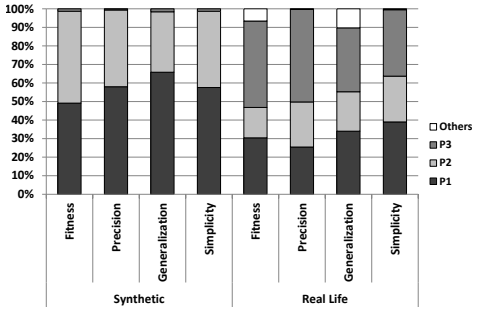
The results of this evaluation study show that `RS4PD` can effectively be used to recommend discovery techniques. The results suggest that the system is highly accurate for most of the event logs. However, there are cases for which the system does not perform so well. This situation can be explained either (i) by the fact the logs are not effectively characterized by the current set of features or (ii) by the lack of experiments on logs characterized by specific features. Eventually, this can be solved by adding other feature extractors to the system. Also, increasing the number of event logs in the system's repository should enhance the quality of the prediction models and, thus, the system's accuracy.

6 Parameters Setting

The selection of parameters for discovery algorithms is considered one of the most challenging issues of this work. The current implementation of the `RS4PD` simply takes into account the default parameters of discovery algorithms when running the experiments (if there are some). However, it is acknowledged that this is a limitation of the recommender system and some approaches were already considered for improving the current work. One simple approach is the instantiation of different versions of the same technique with different values for its parameters, and consider each version as a different algorithm in the recommender system. One of the challenges of this approach is (still) the selection of a good set of instantiations that effectively covers the parameter space. Also, considering multiple instances imply a higher number of experiments to support the recommender system. Another approach is the *parameter optimization* in which parameter space is searched in order to find the best parameters setting with respect to a specific quality measure. The main challenge of this approach is to select a robust strategy to search the parameter space. Traditional strategies such as genetic algorithms have proven to be effective in optimization problems, but they are usually computationally costly. A third approach, which may also be used to facilitate the parameter optimization, is known as *sensibility analysis* and consists of assessing the influence of the inputs of a mathematical model (or system) on the model's output. This information may help on understanding the relationship between the inputs and the output of the model, or identifying redundant inputs in specific contexts. Sensibility methods range from variance-based methods to screening techniques. One of the advantages of screening is that it requires a relatively low number of evaluations when compared to other approaches.

Screening experiments based on the *Elementary Effect* (EE) method [6,3] can be applied to identify non-influential parameters of control-flow algorithms, which usually are computationally costly for estimating other sensitivity analysis measures (e.g., variance-based measures). Rather than quantifying the exact importance of parameters, the EE method provides insight into the contribution of parameters to the results quality.

One of the most efficient EE methods is based on Sobol's quasi-random numbers [11] and a radial OAT strategy [3].[10] The main idea is to analyze the parameter space by performing experiments and assessing the impact of changing parameters with respect to the results quality. A Sobol's quasi-random generator is used to determine a uniformly distributed set of points in the parameter space. Radial OAT experiments [3] are executed over the generated points to measure the impact of the parameters. This information can be used either (i) to guide the users of the `RS4PD` on the parameters setup by prioritizing the parameters to be tunned, or (ii) as a first step towards parameter optimization in the `RS4PD`.



The figure on the right presents the results of a preliminary study about the impact of the parameters of the FHM. Using the testing dataset described in Section 5, several radial OAT experiments were executed to measure the impact of the FHM's parameters on the four quality measures. The results suggest that, although the FHM has seven parameters, it mainly relies on three parameters: *dependency threshold* ($P1$), *relative-to-best threshold* ($P2$), and *all tasks-connected heuristic* ($P3$). For more structured logs (the synthetic logs), the quality of the process model depends mainly on $P1$ and $P2$. For less structured logs (i.e., real-life), other parameters may be needed for improving the quality of the process model.

7 Conclusions and Future Work

This paper describes a recommender system for process discovery using portfolio-based algorithm selection techniques. To the best of our knowledge, it is the first attempt to incorporate machine learning and information retrieval techniques for recommending process discovery algorithms. Also, the approach is very general and allows for the easy incorporation of new techniques, measurements and log features. Due to its continuous learning principle that makes the system to be decoupled in a server-client architecture, the initial promising results obtained are expected to be even better when a larger training set will be available.

As future work, besides the ideas presented in Section 6, several lines will be pursued. First, research is required to improve and extend the current log features. Second, the incorporation of other discovery and conformance techniques will be considered. Third, the encapsulation of the presented recommender system as a pure discovery plugin will be considered, to deliver the user of navigating through the results and thus simplifying the discovery task. Fourth, the incorporation of user-feedback into the training loop will be considered (e.g., usefulness of results or user goals), to improve the usage of the provided recommendations. This feedback may also be used to qualitatively assess the recommender system.

[10] OAT stands for One (factor) At a Time.

References

1. R. Akbarinia, E. Pacitti, and P. Valduriez. Best Position Algorithms for Top-k Queries. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB '07, pages 495–506, 2007.
2. J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. Recommender Systems Survey. *Knowledge-Based Systems*, 46(0):109–132, 2013.
3. F. Campolongo, A. Saltelli, and J. Cariboni. From Screening to Quantitative Sensitivity Analysis. A Unified Approach. *Computer Physics Communications*, 182(4):978–988, 2011.
4. R. Fagin, A. Lotem, and M. Naor. Optimal Aggregation Algorithms for Middleware. In *Proceedings of the Twentieth Symposium on Principles of Database Systems*, PODS '01, pages 102–113, New York, NY, USA, 2001. ACM.
5. M. Mısır and M. Sebag. Algorithm Selection as a Collaborative Filtering Problem. Technical report, INRIA, 2013.
6. M.D. Morris. Factorial Sampling Plans for Preliminary Computational Experiments. *Technometrics*, 33(2):161–174, April 1991.
7. E. O'Mahony, E. Hebrard, A. Holland, C. Nugent, and B. O'Sullivan. Using Case-Based Reasoning in an Algorithm Portfolio for Constraint Solving. In *Irish Conference on Artificial Intelligence and Cognitive Science*, 2008.
8. J.R. Rice. The Algorithm Selection Problem. *Adv. in Computers*, 15:65–118, 1976.
9. A. Rozinat, A.K. A. de Medeiros, C.W. Günther, A.J.M.M. Weijters, and W.M.P. van der Aalst. Towards an Evaluation Framework for Process Mining Algorithms. Technical Report 224, Eindhoven University of Technology, 2006.
10. A. Rozinat and W.M.P. van der Aalst. Conformance Checking of Processes Based on Monitoring Real Behavior. *Information Systems*, 33(1):64–95, March 2008.
11. I.M. Sobol. Uniformly Distributed Sequences With an Additional Uniform Property. *USSR Computational Mathematics and Mathematical Physics*, 16(5):236 – 242, 1976.
12. X. Su and T.M. Khoshgoftaar. A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence*, 2009.
13. W.M.P. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer, Berlin, 2011.
14. S. vanden Broucke, C. Delvaux, J. Freitas, T. Rogova, J. Vanthienen, and B. Baesens. Uncovering the Relationship between Event Log Characteristics and Process Discovery Techniques. In *Proceedings of the 9th Workshop on Business Process Intelligence (BPI 2013)*, 2013.
15. S. vanden Broucke, J.D. Weerdt, B. Baesens, and J. Vanthienen. A Comprehensive Benchmarking Framework (CoBeFra) for conformance analysis between procedural process models and event logs in ProM. In *IEEE Symposium on Computational Intelligence and Data Mining*, Grand Copthorne Hotel, Singapore, 2013. IEEE.
16. H.M.W. Verbeek, J.C.A.M. Buijs, B.F. van Dongen, and W.M.P. van der Aalst. ProM 6: The Process Mining Toolkit. In *Demo at the 8th International Conference on Business Process Management*, volume 615 of *CEUR-WS*, pages 34–39. 2010.
17. J. Wang, R.K. Wong, J. Ding, Q. Guo, and L. Wen. On Recommendation of Process Mining Algorithms. In *2012 IEEE 19th International Conference on Web Services (ICWS)*, pages 311–318, 2012.
18. P. Weber, B. Bordbar, P. Tino, and B. Majeed. A Framework for Comparing Process Mining Algorithms. In *GCC Conference and Exhibition, IEEE*, pages 625–628, 2011.
19. A.J.M.M. Weijters and J.T.S. Ribeiro. Flexible Heuristics Miner (FHM). In *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2011, Paris, France*. IEEE, 2011.
20. L. Xu, F. Hutter, H.H. Hoos, and K. Leyton-Brown. SATzilla: Portfolio-Based Algorithm Selection for SAT. *J. of Artif. Intelligence Research*, 32(1):565–606, 2008.