Heuristic approaches for generating local process models through log projections

Tax, N.; Sidorova, N.; van der Aalst, W.M.P.; Haakma, R.

Published in:
Proceedings of IEEE Symposium on Computational Intelligence and Data Mining, December 6-9, 2016, Athens, Greece

DOI:
10.1109/SSCI.2016.7849948

Published: 01/01/2017

Citation for published version (APA):
Tax, N., Sidorova, N., van der Aalst, W. M. P., & Haakma, R. (2017). Heuristic approaches for generating local process models through log projections. In Proceedings of IEEE Symposium on Computational Intelligence and Data Mining, December 6-9, 2016, Athens, Greece (pp. 1-8). Piscataway: Institute of Electrical and Electronics Engineers (IEEE). DOI: 10.1109/SSCI.2016.7849948

Heuristic Approaches for Generating Local Process Models through Log Projections

Niek Tax, Natalia Sidorova, Wil M. P. van der Aalst
Department of Mathematics and Computer Science
Eindhoven University of Technology
P.O. Box 513, 5600MB Eindhoven, The Netherlands
Email: {n.tax,n.sidorova,w.m.p.v.d.aalst}@tue.nl

Reinder Haakma
Philips Research
Prof. Holstlaan 4, 5665 AA Eindhoven
The Netherlands
Email: reinder.haakma@philips.com

***Abstract*—Local Process Model (LPM) discovery is focused on the mining of a set of process models where each model describes the behavior represented in the event log only partially, i.e. subsets of possible events are taken into account to create so-called local process models. Often such smaller models provide valuable insights into the behavior of the process, especially when no adequate and comprehensible single overall process model exists that is able to describe the traces of the process from start to end. The practical application of LPM discovery is however hindered by computational issues in the case of logs with many activities (problems may already occur when there are more than 17 unique activities). In this paper, we explore three heuristics to discover subsets of activities that lead to useful log projections with the goal of speeding up LPM discovery considerably while still finding high-quality LPMs. We found that a Markov clustering approach to create projection sets results in the largest improvement of execution time, with discovered LPMs still being better than with the use of randomly generated activity sets of the same size. Another heuristic, based on log entropy, yields a more moderate speedup, but enables the discovery of higher quality LPMs. The third heuristic, based on the relative information gain, shows unstable performance: for some data sets the speedup and LPM quality are higher than with the log entropy based method, while for other data sets there is no speedup at all.**

I. Introduction

Process mining is an area of research that combines methods and techniques from computational intelligence, data mining, and process analysis to extract novel insights from event data [1]. One of the most prominent tasks within the process mining field is *process discovery*, where the goal is to discover a process model that accurately describes some sequences – called *traces* – of event data from start to end. In some application domains a high degree of variability can be found in such traces. An example of such a domain is human behavior [2], with events registered e.g. in smart homes, by wearables, or manually in so called LifeLogs. Another application domain, where the variability in the process is often large is the medical workflows of patient care [3].

Existing process discovery algorithms (e.g. [4], [5]) often fail to generate insightful models on such event logs and generate process models in which any sequence of events is allowed, often referred to as the *flower model*. More insight in such event logs can often be obtained using declarative process discovery algorithms (e.g. [6], [7]). Declarative models describe the behavior through a set of constraints on activities (event types), e.g. binary constraint "each activity a is always followed by activity b". They primarily focus on binary constrains, with a limited set of non-binary extensions by branching, like "each activity a is always followed by activity b or activity c". Richer models of unstructured processes can be discovered using *Local Process Model (LPM)* discovery [8]. LPMs describe frequent behavioural patterns in a process modeling notation (e.g. Petri nets, BPMN, UML activity diagrams), allowing each pattern to have full expressive power of the respective process model notation. LPMs allow for complex relations between activities.

Fig. 1 shows some of the LPMs discovered on the log extracted from MIMIC-II [9], a medical database containing 147461 logged medical procedure events from 1734 activities for 28280 patients collected between 2001 and 2008 from multiple intensive care units (ICUs) in the USA. The first LPM indicates that the placement of continuous invasive mechanical ventilation on a patient is 3638 out of 8291 times followed by either arterial catheterization or by one or more instances of infusion of concentrated nutrition. The second LPM shows that the placement of invasive mechanical ventilation frequently co-occurs with the insertion of an endotracheal tube. The third LPM shows that a sequence of one or more hemodialysis events is always followed by the placement of a venous catheter for renal dialysis, and from the numbers we can deduce that there are on average almost 4 hemodialysis events before one placement of a venous catheter. The fourth LPM in Fig. 1 shows that all placements of a continuous invasive mechanical ventilation are eventually followed by the placement of a non-invasive mechanical ventilation. However, on average 8 continuous invasive mechanical ventilations have been applied before the placement of a non-invasive mechanical ventilation.

The LPMs in Fig. 1 cannot be discovered using the brute-force approach described in [8]. There are 1734 activities in the log, yielding an incredible number of models. The computational complexity of LPM discovery grows combinatorially with the number of activities in the event log, hindering the practical application to many real life event logs. However, LPMs can be discovered on a projection on a subset of the activities in the log, as long as the set of activities projected on contains at least the activities in the LPM. The top LPM in Fig. 1 can for example be discovered using a projection on just the activities *CONTINUOUS INVASIVE MECH*, *ARTERIAL CATHETERIZATION*,

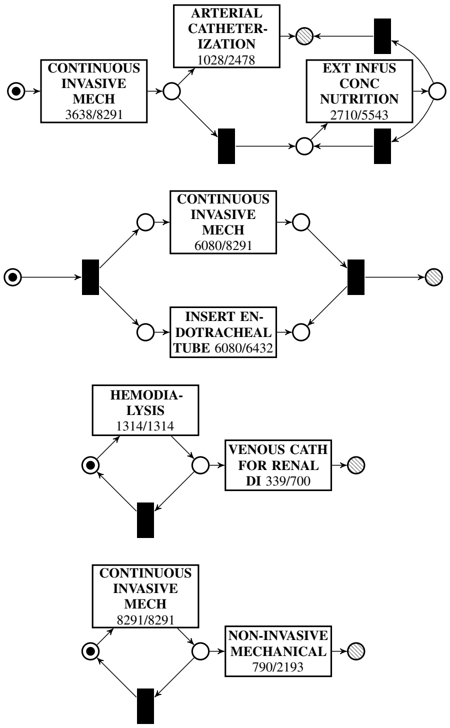

ARTERIAL CATHETER- IZATION 1028/2478
CONTINUOUS INVASIVE MECH 3638/8291
EXT INFUS CONC NUTRITION 2710/5543
CONTINUOUS INVASIVE MECH 6080/8291
INSERT EN- DOTRACHEAL TUBE 6080/6432
HEMODIA- LYSIS 1314/1314
VENOUS CATH FOR RENAL DI 339/700
CONTINUOUS INVASIVE MECH 8291/8291
NON-INVASIVE MECHANICAL 790/2193

Fig. 1. Local process models discovered using projection set discovery on the MIMIC II medical procedure data set

and *EXT INFUS CONC NUTRITION*. In this paper, we explore and compare three heuristic approaches to LPM mining, using intelligently chosen subsets of the activities. By applying LPM discovery to an event log projected on the subsets of activities that are discovered with the heuristic approaches the computational time of discovery decreases considerably. For example, the LPMs in Fig. 1 were discovered in 12 seconds using the Markov clustering approach that we will discuss in Section IV.

We start by introducing basic concepts and notations in Section II. In Section III we discuss quality criteria and rankings of Local Process Models. In Section IV we introduce the three heuristic approaches to discovery of projection sets. We introduce an experimental setup for evaluation of heuristics and describe five real-life event logs used for evaluation in Section V. Section VI contains the results of the experiments and their discussion. We discuss related work in Section VII and conclude this paper in Section VIII.

II. Preliminaries

In this section we introduce concepts used in later sections of this paper.

X^* denotes the set of all sequences over a set X and $\sigma = \langle a_1, a_2, \dots, a_n \rangle$ a sequence of length n; $\langle\rangle$ is the empty sequence and $\sigma_1 \cdot \sigma_2$ is the concatenation of sequences σ_1, σ_2.

In the context of process logs, we assume the set of all *process activities* Σ_L to be given. An *event* e in an event log is the occurrence of an activity $e \in \Sigma_L$. We call a sequence of events $\sigma \in \Sigma_L^*$ a *trace*. An *event log* $L \in \mathbb{N}^{\Sigma_L^*}$ is a finite multiset of traces. For example, the event log $L = [\langle a, b, c\rangle^2, \langle b, a, c\rangle^3]$ consists of 2 occurrences of trace $\langle a, b, c\rangle$ and three occurrences of trace $\langle b, a, c\rangle$. A projection of sequence $\sigma \in X^*$ on a subset $X' \subseteq X$ is denoted by $\sigma\restriction_{X'}$. For example, $\langle a, b, c, a, b, c\rangle\restriction_{\{a,c\}} = \langle a, c, a, c\rangle$. We also lift the projection operator to multi-sets of sequences. For example, $[\langle a, b, c, a, b, c\rangle^3, \langle a, c, a, d, c\rangle^2, \langle a, c, d, c\rangle^4]\restriction_{\{a,c\}} = [\langle a, c, a, c\rangle^5, \langle a, c, c\rangle^4]$.

In the context of process mining, commonly used statistics over the log are related to the activities directly following/preceding each other in the traces of the log [1]. The *directly follows ratio* $dfr(a, b, L)$ for activities (a, b) in log L is the ratio of occurrences of a that are directly followed by b in event log L to the total number of the occurrences of a in L. The *directly precedes ratio* of (a, b) in log L, denoted $dpr(a, b, L)$, is the ratio of occurrences of a directly preceded by a b to the total number of the occurrences of a in L. Assuming an arbitrary but fixed order over the activities of Σ_L, $dpr(a, L)$ and $dfr(a, L)$ represent respectively the vectors of values $dpr(a, b, L)$, $dfr(a, b, L)$ for all $b \in \Sigma_L$.

Petri nets is a process modeling formalism frequently used in process modeling and process mining. A Petri net is a directed bipartite graph consisting of places (depicted as circles) and transitions (depicted as rectangles), connected by arcs. Transitions represent activities, while places represent the enabling conditions of transitions. Labels are assigned to transitions to indicate the type of activity that they model. A special label τ is used to represent invisible transitions (depicted as black rectangles), which are only used for routing purposes and not recorded in the execution log.

Definition 1 (Labeled Petri net): A *labeled Petri net* $N = \langle P, T, F, \Sigma_M, \ell\rangle$ is a tuple where P is a finite set of places, T is a finite set of transitions such that $P \cap T = \emptyset$, $F \subseteq (P \times T) \cup (T \times P)$ is a set of directed arcs, called the flow relation, Σ_M is a finite set of labels representing activities, with $\tau \notin \Sigma_M$ being a label representing invisible events, and $\ell : T \to \Sigma_M \cup \{\tau\}$ is a labeling function that assigns a label to each transition.

For a node $n \in (P \cup T)$ we use $\bullet n$ and $n\bullet$ to denote the set of input and output nodes of n. A state of a Petri net is defined by its *marking* $M \in \mathbb{N}^P$ being a multiset of places. A marking is graphically denoted by putting $M(p)$ tokens on each place $p \in P$. A pair (N, M) is called a marked Petri net. State changes occur through transition firings. A transition t is enabled (can fire) in a given marking M if each input place $p \in \bullet t$ contains at least one token. Once a transition fires, one token is removed from each input place of t and one token is added to each output place of t, leading to a new marking. A firing of a transition t leading from marking M to marking M' is denoted as $M \xrightarrow{t} M'$. $M_1 \xrightarrow{\gamma} M_2$ indicates that M_2 can be reached from M_1 through firing sequence $\gamma \in T^*$.

Often, it is useful to consider a Petri net in combination with an initial marking and a set of possible final markings. This allows us to define the language accepted by the Petri net as a set of finite sequences of activities.

Definition 2 (Accepting Petri Net): An *accepting Petri net* is a triple $APN = (N, M_0, MF)$, where N is a labeled Petri net, $M_0 \in \mathbb{N}^p$ is its initial marking, and $MF \subset \mathbb{N}^p$ is its set of possible final markings, such that $\forall_{M_1,M_2 \in MF}\ M_1 \not\subset M_2$. A sequence $\sigma \in \Sigma_M^*$ is a *trace* of an accepting Petri net APN if there exists a firing sequence $M_0 \xrightarrow{\gamma} M_f$ such that $M_f \in MF$, $\gamma \in T^*$ and $\ell(\gamma) = \sigma$. The *language* $\mathfrak{L}(APN)$ is the set of all its traces, which can be infinite when APN contains one or more loops.

The four models in Fig. 1 are accepting Petri nets. Places that belong to the initial marking contain a token and places belonging to a final marking are simply marked as ◎, since the nets in the figure have a single final marking. The language of the accepting Petri net at the bottom consists of all the sequences starting with one of more *CONTINUOUS INVASIVE MECH* followed by *NON-INVASIVE MECHANICAL.*

III. Local Process Models and Their Rankings

Local Process Model (LPM) discovery generates a set of LPMs that each individually describe some frequent behavior in the event log. The quality of an LPM LN is calculated using segmentation of traces from a log into sequences $\gamma_0\xi_1\gamma_1\xi_2\dots\xi_k\gamma_k$, with $\xi_i \in \mathfrak{L}(LN)$ and $\gamma_i \notin \mathfrak{L}(LN)$, such that the number of events in $\xi_1\dots\xi_k$ is maximized: the higher the number of events in ξ segments, the larger the share of traces of the log explained by the LPM. The ranking of Local Process Models is based on a weighted average over five quality criteria in a zero to one range, as described in [8]:

Support The frequency of the behavior described by the LPM in the event log, i.e. the number of trace segments in the log that fit $\mathfrak{L}(LN)$.

Confidence The share of events of the activities described by the LPM that abide to the behavior described by the LPM, i.e., are in a $\xi_i \in \mathfrak{L}(LN)$ segment.

Language fit The ratio of traces from $\mathfrak{L}(LN)$ that were observed at least once in the event log to $|\mathfrak{L}(LN)|$ (bounded up to a certain length for models with loops).

Determinism This metric reflects the average number of enabled transitions in each marking of the LPM reached while replaying the event log on the LPM. The intuition behind this is that a model with a higher degree of non-determinism captures less information about the ordering of events.

Coverage The frequency of the activities described in the LPM in the event log.

In [8] we developed an incremental procedure for building LPMs, starting from models with two activities, and recursively extending them to more activities. The support and the determinism quality dimensions can be used there for pruning thanks to their monotonicity with respect to model extensions, resulting in speedup of LPM discovery because of a smaller search space of LPMs.

IV. Discovering Log Projection Sets

Local Process Models (LPMs) only contain a subset of the activities of the log and each LPM can in principle be discovered on any projection of the log containing the activities used in this LPM. In this section we describe three heuristics for discovery of *projection sets* – subsets of activities to be used for projecting the log. Each heuristic described takes an event log as input and produces a set of projection sets. These projection sets could potentially be overlapping, which is a desired property as interesting patterns might exist within a set of activities $\{A, B, C, D\}$, as well as within a set of activities $\{A, B, C, E\}$, and discovering on both subsets individually is faster than discovering once on $\{A, B, C, D, E\}$. None of the projection sets in this set is a subset of another projection set to avoid double work, as each LPM that can be discovered on a certain projection set can also be discovered on a superset of that same projection set.

A. An approach based on Markov clustering

Markov clustering [10], [11] is a fast and scalable clustering algorithm for graphs that is based on simulation of flow in graphs. The main intuition behind Markov clustering is that, while performing a random walk on a graph, the likelihood of transitioning between two members of the same cluster is higher than the likelihood of transitioning between two nodes that are in different clusters. Markov clustering takes as input a Markov matrix, i.e. a matrix that describes the transition probabilities in a Markov chain. We generate a matrix M that represents the *connectedness* of two activities by using the directly-follows and directly-precedes ratios: $M_{i,j} = \sqrt{dpr(i,j,L)^2 + dfr(j,i,L)^2}$, using the L_2 norm of the directly precedes ratio and the directly follows ratio. A Markov matrix M' is obtained by normalizing M row-wise. The intuition behind using both *dpr* and *dfr* combined instead of either of the two is that it can be both of importance that activity i is often followed by j and that j is often preceded by i; if either of the two is true than there is apparently some relation $i \to j$. Applying Markov clustering to M' results in a set of clusters of activities, where we use each activity cluster as a projection set. Markov clustering simulates a random walk over a graph by alternating *expansion*, taking the power of this matrix, and *inflation*, taking the entrywise power of this matrix. The clusters generated by Markov clustering can overlap, which is a desired property in projection set discovery. The inflation parameter of the Markov clustering algorithm is known to be the main parameter in determining the granularity of the clustering obtained [11] with Markov clustering.

B. Log entropy based approach

Another approach to generate projections sets is to form groups of activities such that the categorical probability distributions over activities preceding or following an activity in the projected logs are peaked, i.e. far away from the discrete uniform distribution. When after an occurrence of activity a observing each other type of activity as the next event in the projected log is equally likely, the activities can be considered to be independent of each other. If, on the other hand, the occurrence of activity a conveys information on the next event and e.g. makes it very likely that some other activity b is going to be observed next, the activities in the projection set are likely to be somehow related.

We use the standard entropy function over categorical probability distribution X over $|X|$ elements: $H(X) = \sum_{x \in X} -x \cdot log_2(x)$. We calculate the total entropy $\mathit{Ent}(L)$ of the log statistics in original event log L as:

$$\mathit{Ent}(L) = \sum_{a \in \Sigma_L} (H(\mathit{dfr}(a, L)) + H(\mathit{dpr}(a, L))).$$

We choose an entropy ratio threshold parameter r to indicate the maximum entropy of the log statistics of a log projection. We start from a set of elementary log projection sets, $S_1 = \{\{a\} | a \in \Sigma_L\}$. We proceed iteratively as follows: we define $S_{i+1} = \{A \cup B | A \in S'_i, B \in S_1 : B \nsubseteq A\}$ and select those projection sets of S_{i+1} that lead to the log projections whose total entropy does not exceed the threshold defined by parameter r: $S'_{i+1} = \{A | A \in S_{i+1} \wedge \mathit{Ent}(L\upharpoonright_A) \leq r \cdot \mathit{Ent}(L)\}$. The procedure stops if $S'_{i+1} = \emptyset$ or $S_{i+1} = \Sigma_L$. Since LPMs that can be discovered using some projection set can also be discovered using a superset of this projection set, projection sets that are subsets of other projection sets are removed from the final result: $S_{\mathit{final}} = \{A | \exists S'_i : (A \in S'_i \wedge (\forall S'_j, \forall B \in S'_j : A \not\subset B))\}$.

C. Maximal Relative Information Gain based approach

A more local perspective on entropy-based projection set discovery would be to compare a projection set with the projection set of the previous time step, instead of the original log. We add an activity to a projection set when adding it strongly decreases the entropy of at least one of the categorical probability distributions over following or preceding activities, even when the entropy of other categorical probability distributions might increase. The intuition behind this is that any decrease in entropy of a categorical probability distribution over following or preceding activities indicates that some pattern is getting stronger by adding that activity and that a LPM could potentially be found.

We define the Maximal Relative Information Gain (MRIG) of projection set A over projection set B, with $A \supset B$, on event log L, $\mathit{MRIG}(A, B, L)$, as the maximal relative information gain over all the log statistics on L, that is the ratio of bits for encoding the log statistic that is most decreased by growing the projection set: $\mathit{MRIG}(A, B, L) = \max_{a \in B} max\{\frac{H(\mathit{dfr}(a, L\upharpoonright_B)) - H(\mathit{dfr}(a, L\upharpoonright_A))}{H(\mathit{dfr}(a, L\upharpoonright_B))}, \frac{H(\mathit{dpr}(a, L\upharpoonright_B)) - H(\mathit{dpr}(a, L\upharpoonright_A))}{H(\mathit{dpr}(a, L\upharpoonright_B))}\}$.

We choose a threshold parameter r to indicate the minimum value of MRIG for considering a projection set as potentially interesting. Like in the log entropy based approach, we start from the set of elementary log projection sets, $S_1 = \{\{a\} | a \in \Sigma_L\}$ and $S'_1 = S_1$. We proceed iteratively defining $S_{i+1} = \{A \cup B | A \in S'_i, B \in S_1 : B \nsubseteq A\}$ and $S'_{i+1} = \{A | A \in S_{i+1} \wedge \exists B \in S'_i : \mathit{MRIG}(A, B, L) > r\}$, filtering out those projection sets where adding an extra activity to the projection set leads to an insufficient decrease in the number bits of entropy needed to encode the log statistics. The procedure stops if $S'_{i+1} = \emptyset$ or $S_{i+1} = \Sigma_L$. Again, projection sets that are subsets of other projection sets are finally removed: $S_{\mathit{final}} = \{A | \exists S'_i : A \in S'_i \wedge (\forall S'_j, \forall B \in S'_j : A \not\subset B)\}$.

V. Experimental setup

Fig. 2 gives an overview of the methodology for evaluation of projection set discovery methods. We assess their quality via the comparison of the quality of the Local Process Models (LPMs) discovered using the projection sets that they generate with the quality of the LPMs discovered on the same log without the use of projections. First, we apply LPM discovery to the original, unprojected, event log L, resulting in the "ideal" top k of LPMs. Then we apply one of the projection set discovery methods on event log L, resulting in a set Q of projection sets. On each of the projected event logs in $L\upharpoonright_Q$ we apply LPM Discovery, resulting in $|Q| \times k$ LPMs. We select from them k unique best LPMs in terms of weighted average over the quality criteria. We compare the "ideal" set of LPMs with the set discovered using projections. Theoretically, they can coincide, or be equally good, if for every "ideal" LPM there is a projection set containing its activities. If it is not the case, some of the best scoring models will be missing and we compare how close are the scores of the models present in the set to the scores of the "ideal" LPMs.

Even with the loss of quality in LPM, projections could be a good option to opt for, since they allow to significantly improve the time performance and make LPM discovery possible for logs with many different activities. We evaluate how our projection discovery methods perform compared to random projections as follows: We create a set Q' of random projection sets, consisting of $|Q|$ projection sets where for each projection set $q \in Q$ we create a random projection set $q' \subseteq \Sigma_L$ of the same size as q: $|q'| = |q|$. We apply LPM discovery on each of the projected event logs in $L\upharpoonright_{Q'}$ and select k best LPMs from the discovered ones. The projection discovery method works well if the LPMs from top k score closer to the "ideal" top k than the LPMs generated based on the random projection sets. To obtain statistically relevant results, we create the random projection sets ten times.

A. Metrics for Comparison of Local Process Model Rankings

We define the recall, denoted *recall@k*, as the fraction of LPMs from the "ideal" top k that are also discovered with the use of projections. A more nuanced way of comparison is to look at the weighted average scores of the LPMs in the "ideal" ranking. The intuition behind this is that not being able to find LPMs from the "ideal" top k is less severe in case the alternatively found LPMs are also of good quality, i.e. just below the top k. Furthermore, missing the best-scoring LPMs is more severe than missing the lower scoring LPMs. For this reason, we use Normalized Discounted Cumulative Gain (NDCG@k) [12], [13], one of the most widely used metrics for evaluation of a ranking with an "ideal" ranking in the field of Information Retrieval [14], that gives more weight to the top of the rankings than to the lower parts of the rankings.

Discounted Cumulative Gain (DCG) aggregates the relevant scores of the individual LPMs in the ranking of LPMs in such a

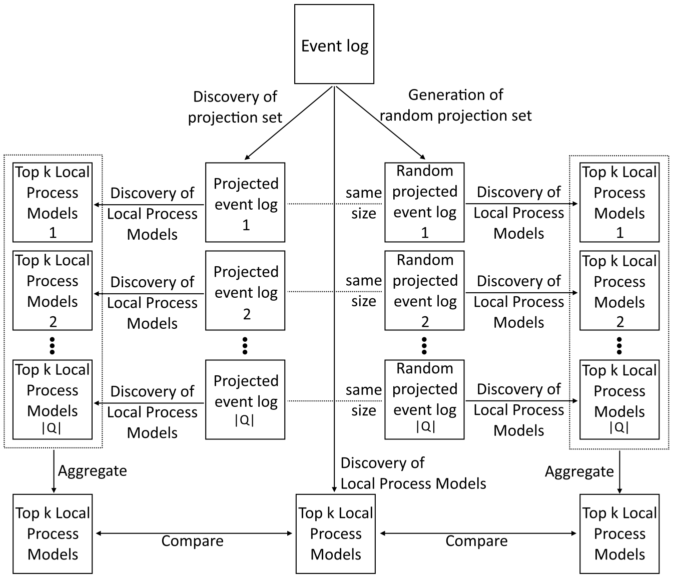

Event log
Discovery of projection set
Generation of random projection set
Top k Local Process Models 1
Discovery of Local Process Models
Projected event log 1
same size
Random projected event log 1
Discovery of Local Process Models
Top k Local Process Models 1
Top k Local Process Models 2
Discovery of Local Process Models
Projected event log 2
same size
Random projected event log 2
Discovery of Local Process Models
Top k Local Process Models 2
Top k Local Process Models |Q|
Discovery of Local Process Models
Projected event log |Q|
same size
Random projected event log |Q|
Discovery of Local Process Models
Top k Local Process Models |Q|
Aggregate
Discovery of Local Process Models
Aggregate
Top k Local Process Models
Compare
Top k Local Process Models
Compare
Top k Local Process Models

Fig. 2. Evaluation methodology for projection set discovery approaches

way that the graded relevance is reduced in the logarithmic proportion to the position of the result; due to that, more weight is put on the top of the ranking than on the lower parts of the ranking. DCG is formally defined as: $DCG@k = \sum_{i=1}^{k} \frac{2^{rel_i}-1}{log_2(i+1)}$, where rel_i is defined as the relevance, which is the weighted average of the model's scores on the quality dimensions for the LPM on position i in the ranking. Normalized Discounted Cumulative Gain (NDCG) is obtained by dividing the DCG value by the theoretical maximum of the discounted cumulative gain value, which is called Ideal Discounted Cumulative Gain (IDCG). The IDCG value is the DCG value obtained from the ground truth ranking on the LPMs discovered on the original, non-projected log, since all local process models that can be discovered from projected event logs can also be discovered from the original event log. Normalized Discounted Cumulative Gain (NDCG) is defined as:

$NDCG@k = \frac{DCG@k}{IDCG@k}$.

B. Data Sets for Projection Discovery Experiments

We perform the evaluation on five different data sets using the methodology described above. All of the data sets used originate from the human behavior logging domain, with four data sets consisting of *Activities of Daily Life* (ADL) and one consisting of activities performed by an employee in a working environment. Event logs from the human behavior domain are generally too unstructured to allow for discovery of informative process models with process discovery techniques, while LPM discovery allows to discover some relations between activities at a local level that are not discoverable with usual process mining techniques. Table I gives an overview of main event log characteristics used in the evaluation. The event logs used for in the experiments have limited number of activities (at most 14), which allows us to determine the ground truth ranking of LPMs, that is, the ranking of LPMs obtained by LPM discovery on the full log without using projections. However, log projections also enable discovery of LPMs on datasets with many more activities, such as the LPMs in Fig. 1 that were discovered on a log with 1734 activities. For the Van Kasteren data set [15] we show and discuss the ranking of LPMs for which we aim to speed up discovery through projections. For each data set we identified a support threshold used for pruning that allows us to run LPM discovery on the unprojected event log within reasonable time (max. 10 minutes on a 4-core 2.4 GHz CPU). This value depends on the number of activities as well as the length of the traces within the event logs, i.e. more activities and/or longer traces result in a need for a higher support threshold to finish the experiment within the time limit.

TABLE I
AN OVERVIEW OF THE DATA SETS USED IN THE EVALUATION EXPERIMENT

Data set	# of activities	# of cases	# of events
BPI '12 resource 10939	14	49	1682
Bruno	14	57	553
CHAD subject 1900010	10	26	238
Ordonez A	12	15	409
Van Kasteren	14	23	1285

1) BPIC '12 Resource 10939 Data Set: The Business Process Intelligence Challenge (BPIC)'12 data set originates from a personal loan application process in a global financial institution. We transformed the event log to obtain traces of daily activities of one specific employee. We set the pruning parameter to 0.675 for this data set.

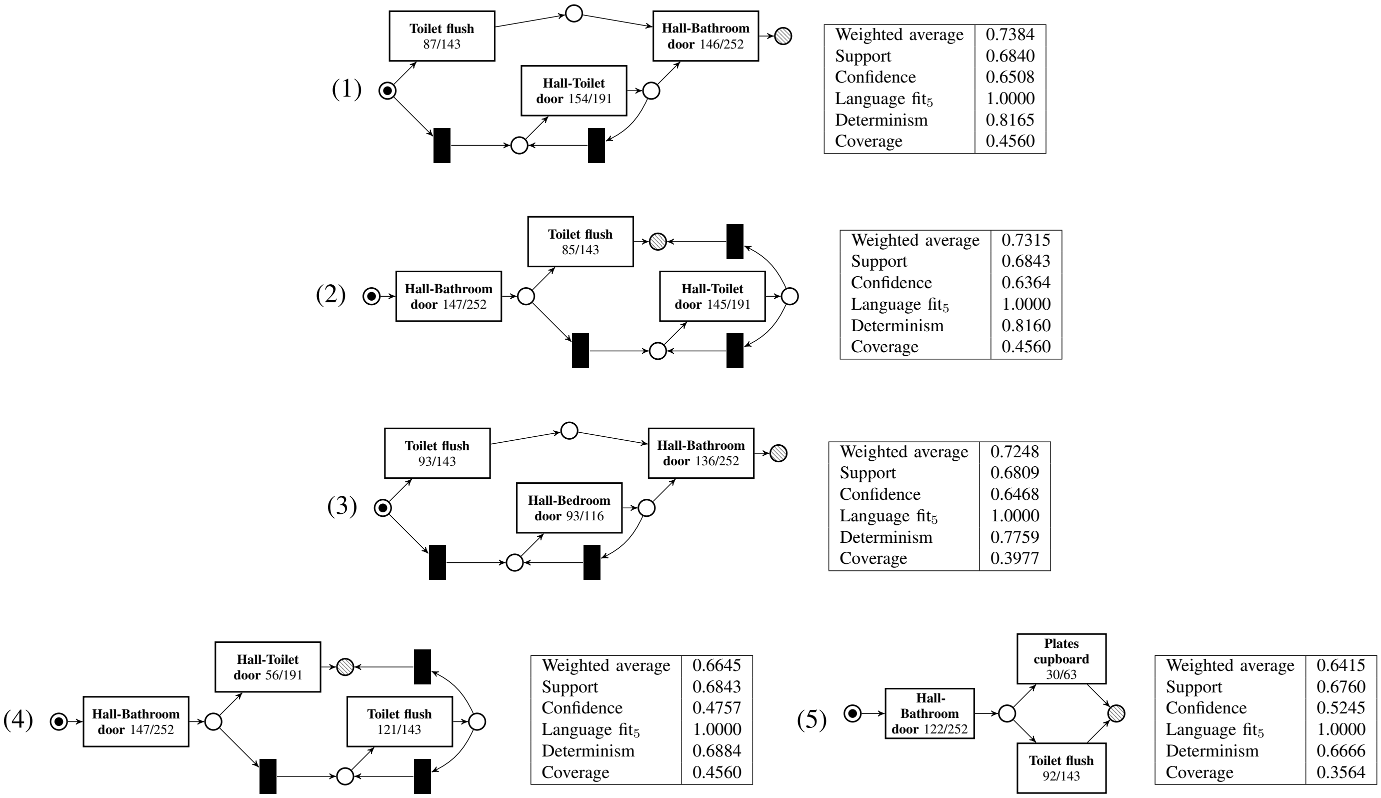
2) Bruno Data Set: The Bruno et al. [16] data set is a public collection of labeled wrist-worn accelerometer recordings. The data set is composed of fourteen types of ADL events performed by sixteen volunteers. We set the pruning parameter to 0.6 for this data set.

3) CHAD Data Set: The CHAD database [17] consists of 22 exposure and time-use studies that have been consolidated in a consistent format. In total the database contains 54000 individual days of human behavior, from which we extract an event log from a randomly chosen study subject such that each case represents a day. For this data set we set the support pruning parameter to 0.4.

4) Ordonez A Data Set: The Ordonez [18] data set consists of ADL events that are performed by two users in their own homes, recorded through smart home sensors. We use an event log obtained from sensor events of subject A, with each case representing a day. We set the pruning parameter to 0.6.

5) Van Kasteren Data Set: This data set is a smart home environment event log described by Van Kasteren et al. [15]. It consists of multidimensional time series data, where each dimension represents the binary state of an in-home sensor. These sensors include motion sensors, open/close sensors, and power sensors (discretized to on/off states). We transform the multidimensional sensor time series into events by considering the change points of sensor states as events. We create cases by grouping events by day, with a cut-off point at midnight. We set the pruning parameter to 0.675 for this data set.

Fig. 3 shows the first five elements of the Local Process Model ranking discovered on one of the data sets (Van Kasteren [15]). The 5^{th} LPM shows that roughly half of the times that



(1)

Weighted average	0.7384
Support	0.6840
Confidence	0.6508
Language fit$_5$	1.0000
Determinism	0.8165
Coverage	0.4560

(2)

Weighted average	0.7315
Support	0.6843
Confidence	0.6364
Language fit$_5$	1.0000
Determinism	0.8160
Coverage	0.4560

(3)

Weighted average	0.7248
Support	0.6809
Confidence	0.6468
Language fit$_5$	1.0000
Determinism	0.7759
Coverage	0.3977

(4)

Weighted average	0.6645
Support	0.6843
Confidence	0.4757
Language fit$_5$	1.0000
Determinism	0.6884
Coverage	0.4560

(5)

Weighted average	0.6415
Support	0.6760
Confidence	0.5245
Language fit$_5$	1.0000
Determinism	0.6666
Coverage	0.3564

Fig. 3. Example top 5 Local Process Models discovered from the Van Kasteren data set.

the *Hall-Bathroom door* is opened, either *Toilet flush* or *Plates cupboard* are observed afterwards. The bathroom in this house contains both a toilet and a shower, but showering events are not observed as the shower does not contain a sensor. It is likely that *Hall-Bathroom* events that are followed by *Plates cupboard* events represent a morning shower after which the subject proceeds his morning routine with breakfast. The ordering of LPMs matches the weighted average of the LPM quality criteria.

VI. Results & Discussion

Fig. 4 shows the results of the evaluation using the five evaluation data sets and Table II shows the speedup of projection-based LPM discovery. The speedup mainly depends on the size of the projection sets. Note that we do not compare the computation time of the discovered and the randomly generated projections, as both consist of equally sized projections. The time needed for discovering the projection set is included in the computation time shown in Table II, however, the projection set discovery time is negligible compared to the time needed for LPM discovery. Each dark gray bar in Fig. 4 represents the performance on the measure indicated by the column of the projection set discovery method indicated by the row on the dataset indicated below. Each gray bar indicates the average performance over ten random projection sets of the same size as the discovered projection set belonging to the dark gray bar to its left. The error bars indicate the Standard Error (SE), indicating the uncertainty of the mean performance of the random projection sets.

Our first observation is that all three projection set discovery methods are better than random projections on almost all data sets. The only exception is the Markov-based projection set discovery approach in case of the Bruno data set, which performs worse than random projections, with none of the LPMs discovered on the original data set being found on the projection sets. The Markov clustering based approach to projection set discovery achieves the highest speedup on three of the five data sets, and second highest speedup on the other two data sets. The high speedup indicates that the projections generated by Markov clustering are relatively small, which also explains the quality loss of the LPMs with regard to the ground truth which is typically larger than with the other projection set discovery methods. The size of the clusters created with Markov clustering can be influenced through its inflation parameter, which we set to 1.5. We found lower values to result in all activities being clustered into one single cluster, meaning that the only projection created is the original log itself.

The entropy based approach shows a higher gain than the Markov based approach on the NDCG@{5,10,20} metrics for all data set, when compared to random projections of the same size. However, the obtained speedup of LPM discovery with the entropy based approach is lower than speedup with the Markov based approach on all but one data set.

The MRIG based approach shows significant improvements on all metrics on the BPI'12 and Bruno data sets. On these two data sets it also results in the second highest and highest

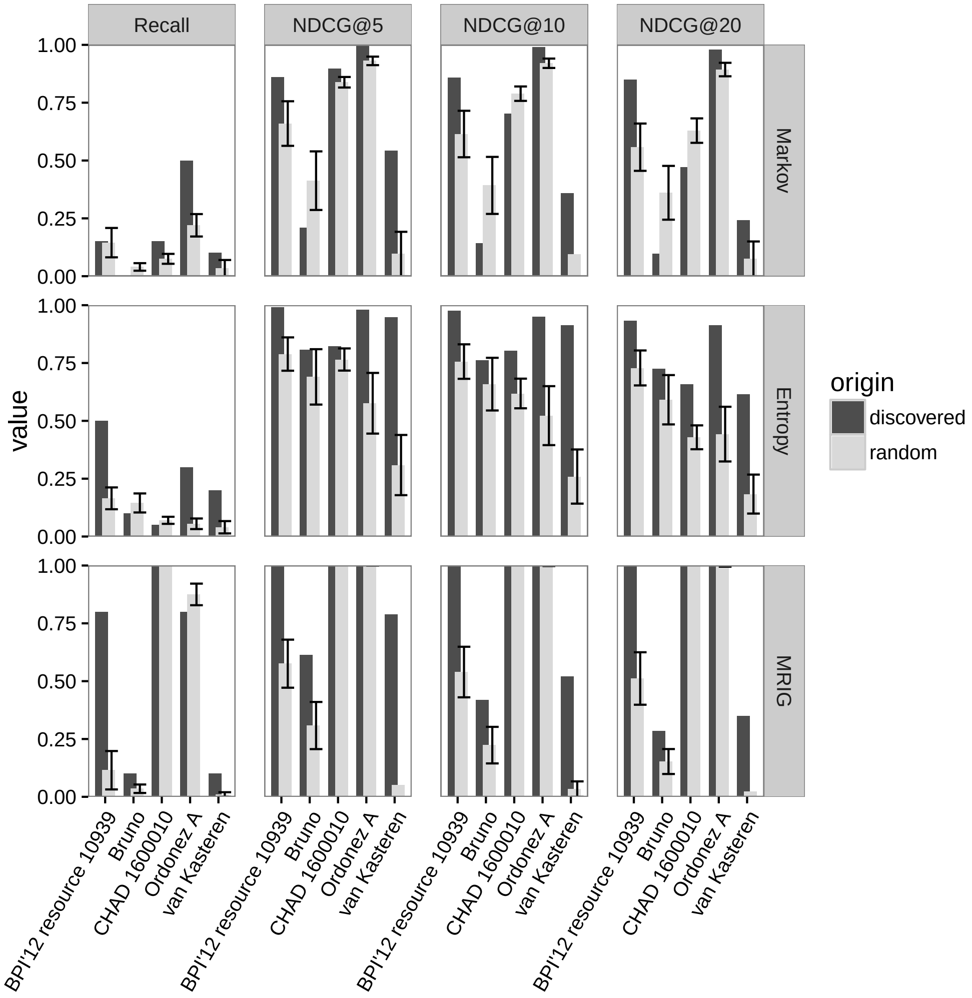


Fig. 4. Performance of the three projection set discovery methods on the six data sets on the four metrics

TABLE II
LOCAL PROCESS MODEL DISCOVERY SPEEDUP OBTAINED WITH EACH PROJECTION SET DISCOVERY METHOD ON THE EVALUATION DATA SETS

Data set	Speedup Markov	Speedup entropy	Speedup MRIG
BPI’12 resource 10939	**42.9**	5.6	6.9
Bruno	222.9	111.5	**336.1**
CHAD subjection 1900010	**1060.1**	1041.2	1
Ordonez A	7.3	**33.7**	1.1
Van Kasteren	**111.2**	83.5	96.7

speedup respectively. However, on the CHAD data and the Ordonez data the discovered projection sets consist of a single projection that contains almost all log activities, resulting consequently in no speedup with close to perfect recall scores.

VII. RELATED WORK

The task of discovering projections plays an important role within the area of decomposed process discovery and conformance checking. Decomposed process discovery aims at partitioning the activities in the event log such that after applying process discovery to each partition of the events, the start-to-end process model can be constructed by stitching together the process models discovered on the individual partitions. In [19] an approach was introduced to decompose process mining by using a *maximal decomposition* of a causal dependency graph, where the activities associated with each edge in the causal dependency graph end up in one cluster. Hompes et al. [20] describe an approach to make more coarse-grained activity clusters by recombining the clusters of the maximal decomposition by balancing three quality criteria: cohesion, coupling, and balance. Van der Aalst and Verbeek [21] introduced a decomposed process mining approach based on *passages*. A passage is a pair of two non-empty sets of activities (X,Y) such that the set of direct successors of X is Y and the set of direct predecessors of Y is X. Munoz-Gama et al. [22] proposed a decomposed conformance checking approach that discovers clusters of activities based on identifying Single-Entry Single-Exit (SESE) blocks in a Petri net model of the process. A SESE block in a Petri net is a set of edges that has exactly two boundary nodes: one entry and one exit. Clustering activities using this approach assumes availability of a structured process model describing process instances from the beginning to the end, which is different from the application area of LPM discovery.

Carmona et al. [23] describe an approach to generate overlapping sets of activities from a causal dependency graph and uses it to speed up region theory based Petri net synthesis from a transitions system representation of the event log. The activities are grouped together in such a way that the synthesized Petri nets can be combined into a single Petri net generating all the traces of the log. In a more recent paper Carmona [24] describes an approach to discover a set of projections from an event log based on Principle Component Analysis (PCA).

All the projection discovery methods mentioned above aim at the discovery of models that can be combined into a single process model. In the context of LPM discovery, we are not interested in combining process models into one single process model; instead, each LPM is assumed to convey interesting information about a relationship between activities itself. Projection methods in the context of decomposed process mining all aim to minimize overlap between projections, leaving only the overlap needed for combining the individual process models into one. In our case, overlap between clusters is often desired, as interesting patterns might exist within a set of activities $\{A, B, C, D\}$, as well as within a set of activities $\{A, B, C, E\}$, and discovering on both subsets individually is faster than discovering once on $\{A, B, C, D, E\}$. The three projection set discovery methods introduced in this paper exploit this and aim for overlapping projection sets.

The episode miner [25] is a related technique to LPM discovery, which discovers patterns that are less expressive (i.e. limited to partial orders), but is computationally less expensive. While the need for heuristic techniques to speed up episode mining is limited compared to LPM discovery, in principle the three heuristics described to speedup LPM discovery could also be used to speedup the discovery of frequent episodes

VIII. Conclusion & Future Work

We explored three different heuristics for the discovery of projection sets for speeding up Local Process Model (LPM) discovery. These heuristics enable the discovery of LPMs from event logs where it is computationally not feasible to discover LPMs from the full set of activities in the log. All three of them produce better than random projections on a variety of data sets. Projection discovery based on Markov clustering leads to the highest speedup, while higher quality LPMs can be discovered using a projection discovery based on log statistics entropy. The Maximal Relative Information Gain based approach to projection discovery shows unstable performance with the highest gain in LPM quality over random projections on some event logs, while not being able to discover any projection smaller than the complete set of activities on some other event logs. In fact, we would like to explore event log properties that can serve as a predictor for the relative performance of these methods.

References

[1] W. M. P. van der Aalst, *Process Mining: Data Science in Action*. Springer, 2016.

[2] N. Tax, E. Alasgarov, N. Sidorova, and R. Haakma, "On generation of time-based label refinements," in *Proceedings of the 25th International Workshop on Concurrency, Specification and Programming*. CEUR-WS.org, 2016, pp. 25–36.

[3] R. S. Mans, M. H. Schonenberg, M. Song, W. M. P. van der Aalst, and P. J. M. Bakker, "Application of process mining in healthcare – a case study in a dutch hospital," in *International Joint Conference on Biomedical Engineering Systems and Technologies*. Springer, 2008, pp. 425–438.

[4] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Discovering block-structured process models from event logs-a constructive approach," in *International Conference on Applications and Theory of Petri Nets and Concurrency*. Springer Berlin Heidelberg, 2013, pp. 311–329.

[5] J. M. E. M. van der Werf, B. F. van Dongen, C. A. J. Hurkens, and A. Serebrenik, "Process discovery using integer linear programming," in *International Conference on Applications and Theory of Petri Nets*. Springer, 2008, pp. 368–387.

[6] F. M. Maggi, A. J. Mooij, and W. M. P. van der Aalst, "User-guided discovery of declarative process models," in *Computational Intelligence and Data Mining, 2011 IEEE Symposium on*. IEEE, 2011, pp. 192–199.

[7] S. Schönig, C. Cabanillas, S. Jablonski, and J. Mendling, "Mining the organisational perspective in agile business processes," in *International Conference on Enterprise, Business-Process and Information Systems Modeling*. Springer, 2015, pp. 37–52.

[8] N. Tax, N. Sidorova, R. Haakma, and W. M. P. van der Aalst, "Mining Local Process Models," *ArXiv:1606.06066*, 2016.

[9] M. Saeed, C. Lieu, G. Raber, and R. G. Mark, "MIMIC II: a massive temporal ICU patient database to support research in intelligent patient monitoring," in *Computers in Cardiology, 2002*. IEEE, 2002, pp. 641–644.

[10] S. van Dongen, "Graph clustering via a discrete uncoupling process," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 1, pp. 121–141, 2008.

[11] ——, "Graph clustering by flow simulation," Ph.D. dissertation, University of Utrecht, Utrecht, May 2000.

[12] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of IR techniques," *ACM Transactions on Information Systems*, vol. 20, no. 4, pp. 422–446, 2002.

[13] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," in *Proceedings of the 22nd International Conference on Machine Learning*. ACM, 2005, pp. 89–96.

[14] N. Tax, S. Bockting, and D. Hiemstra, "A cross-benchmark comparison of 87 learning to rank methods," *Information Processing & Management*, vol. 51, no. 6, pp. 757–772, 2015.

[15] T. van Kasteren, A. Noulas, G. Englebienne, and B. Kröse, "Accurate activity recognition in a home setting," in *Proceedings of the 10th International Conference on Ubiquitous Computing*. ACM, 2008, pp. 1–9.

[16] B. Bruno, F. Mastrogiovanni, A. Sgorbissa, T. Vernazza, and R. Zaccaria, "Analysis of human behavior recognition algorithms based on acceleration data," in *Robotics and Automation, 2013 IEEE International Conference on*. IEEE, 2013, pp. 1602–1607.

[17] T. McCurdy, G. Glen, L. Smith, and Y. Lakkadi, "The national exposure research laboratory's consolidated human activity database," *Journal of Exposure Analysis and Environmental Epidemiology*, vol. 10, no. 6, pp. 566–578, 2000.

[18] F. J. Ordónez, P. de Toledo, and A. Sanchis, "Activity recognition using hybrid generative/discriminative models on home environments using binary sensors," *Sensors*, vol. 13, no. 5, pp. 5460–5477, 2013.

[19] W. M. P. van der Aalst, "Decomposing Petri nets for process mining: A generic approach," *Distributed and Parallel Databases*, vol. 31, no. 4, pp. 471–507, 2013.

[20] B. F. A. Hompes, H. M. W. Verbeek, and W. M. P. van der Aalst, "Finding suitable activity clusters for decomposed process discovery," in *Data-Driven Process Discovery and Analysis*. Springer, 2014, pp. 32–57.

[21] W. M. P. van der Aalst and H. M. W. Verbeek, "Process discovery and conformance checking using passages," *Fundamenta Informaticae*, vol. 131, no. 1, pp. 103–138, 2014.

[22] J. Munoz-Gama, J. Carmona, and W. M. P. Van Der Aalst, "Single-entry single-exit decomposed conformance checking," *Information Systems*, vol. 46, pp. 102–122, 2014.

[23] J. Carmona, J. Cortadella, and M. Kishinevsky, "Divide-and-conquer strategies for process mining," in *Business Process Management*. Springer, 2009, pp. 327–343.

[24] J. Carmona, "Projection approaches to process mining using region-based techniques," *Data Mining and Knowledge Discovery*, vol. 24, no. 1, pp. 218–246, 2012.

[25] M. Leemans and W. M. P. van der Aalst, "Discovery of frequent episodes in event logs," in *Data-Driven Process Discovery and Analysis*. Springer, 2014, pp. 1–31.