Prototype Selection using Clustering and Conformance Metrics for Process Discovery

Mohammadreza Fani Sani[1], Mathilde Boltenhagen[2], and Wil van der Aalst[1]

[1] RWTH Aachen University, Aachen, Germany
{fanisani, wvdaalst}@pads.rwth-aachen.de
[2] LSV, Université Paris-Saclay, ENS Paris-Saclay, CNRS, Inria, Cachan (France)
{boltenhagen}@lsv.fr

Abstract. Automated process discovery algorithms aim to automatically create process models based on event data that is captured during the execution of business processes. These algorithms usually tend to use all of the event data to discover a process model. Using all (i.e., less common) behavior may lead to discover imprecise and/or complex process models that may conceal important information of processes. In this paper, we introduce a new incremental prototype selection algorithm based on the clustering of process instances to address this problem. The method iteratively computes a unique process model from a different set of selected prototypes that are representative of whole event data and stops when conformance metrics decrease. This method has been implemented using both ProM and RapidProM. We applied the proposed method on several real event datasets with state-of-the-art process discovery algorithms. Results show that using the proposed method leads to improve the general quality of discovered process models.

Keywords: Process Mining · Process Discovery· Prototype Selection · Trace Clustering · Event Log Preprocessing · Quality Enhancement

1 Introduction

Process Mining bridges the gap between traditional data science techniques and business process management analysis [1]. Process discovery, one of the main branches of this field, aims to discover process models (e.g., Petri nets or BPMN) that describe the underlying processes captured within the event data. Event data that is also referred to as *event logs*, readily available in most current information systems [1]. Process models capture choice, concurrent, and loop behavior of activities.

To measure the quality of discovered process models, four criteria have been presented in the literature, i.e., *fitness*, *precision*, *generalization*, and *simplicity* [2]. *Fitness* indicates how much of the observed behavior in data is described by the process model. In opposite, *Precision* computes how much modeled behavior exists in the event log. Generalization represents the ability of a model to correctly capturing parts of the system that have not been recorded [3]. Simplicity measures the understandability of a process model by limiting the number of nodes and complex structures of the resulting model.

Several automated process discovery algorithms have been proposed in the literature that work perfectly on synthetic event logs. However, when dealing with real event logs, many of them have difficulties to discover proper models and generate spaghetti-like process models, i.e., the discovered models contain too many nodes and arcs. Such structures are too complex for human analysis. Therefore, the quality of discovered process models depends on the given event log which can be noisy or very complex [4]. Moreover, sometimes the discovered models are unacceptably imprecise and describe too much behavior compared to the given event log. Thus, many state-of-the-art process discovery algorithms have difficulties to balance between these four quality criteria.

The mentioned problems are usually caused by high data variability of event logs and the existence of infrequent behavior in them. Therefore, by applying *data preprocessing* techniques, e.g., noise reduction [5,6], we are able to decrease the data variability of event logs and consequently improve the resulting process models. Using this approach, the preprocessed event log is given to process discovery algorithms instead of the original event log.

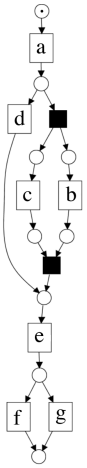
In this paper, we aim to improve the results of process discovery algorithms by proposing a new preprocessing method that incrementally selects prototypes in event logs. Our main motivation is to get the most representable trace instances. For this purpose, the method uses trace clustering. Each cluster has a representative instance that we consider as a *prototype*. The selection of prototypes is incremental and depends on the moderate use of conformance checking artifacts. By using prototypes we reduce the data variability of event logs and consequently improve the precision and simplicity of discovered models.

Using `RapidProM` [7], we study the usefulness of the proposed method by applying it on several real event logs while using different process discovery algorithms. The experimental results show that applying our method improves the balance between the quality metrics of discovered process models.

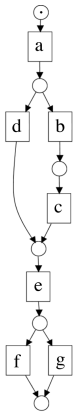
The remainder of this paper is structured as follows. We first provide a motivating example in Section 2. Then, in Section 3, we discuss related work. Section 4 defines preliminary notations. We present the prototype selection method in Section 5. The evaluation and its results are given in Section 6. Finally, Section 7 concludes the paper and presents some future work.

2 Motivating Example

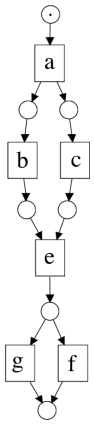
Research like [8] has shown that by using only a small subset of traces for process discovery we sometimes can improve the quality of process models. The main challenge faced this research is which traces should be selected as input for process discovery algorithms. Some methods, e.g.,[9,8], propose to use sampling methods for this purpose without considering the quality of discovered model during the selection phase. We aim to find the most representative process instances of a log, i.e., referred to prototypes, using a clustering method. To motivate our approach, in Fig 1, we show discovered models based on selected traces of an event log (i.e., Fig. 1e) by the inductive miner [10] in conjunction with three preprocessing methods.



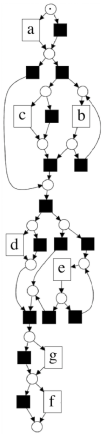
(a) Prototype selection



(b) Biased sampling [8]



(c) Frequency based selection



(d) No preprocessing

$[\langle a,b,c,e,g\rangle^4,\ \langle a,c,b,e,g\rangle^4,$
$\langle a,b,c,e,f\rangle^3,\ \langle a,c,b,e,f\rangle^2$
$\langle a,d,e,f\rangle,\ \langle a,d,e,g\rangle,\ \langle a,b\rangle,$
$\langle a,c,b,f\rangle,\ \langle b,c,e,f\rangle, \langle a,f\rangle,$
$\langle a,b,b,e,d\rangle,\ \langle a,d,e,g,f\rangle,$
$\langle a,b,c,e,e,f\rangle]$

(e) Event log L

Preprocessing Method	Fitness	Precision	F1-Measure
Prototype selection	0.906	1.000	0.951
Biased sampling	0.842	1.000	0.914
Frequency based selection	0.842	1.000	0.914
No preprocessing	1.000	0.701	0.824

(f) Conformance metrics comparison

Fig. 1: Comparison of different trace selection methods using the inductive miner

Note that the statistical sampling method [9] returns all the traces to have a high confidence of not loosing information (and because of the small size of the log). The biased sampling method [8] takes as input the percentage of desired traces. In this example, we used 30% of the entire log to get the same number of traces as our method. Moreover, the frequency based selection method returns the top most frequent traces in the log. Considering conformance metrics of discovered models using different preprocessing methods that is presented in Fig. 1f, we found that choosing the right instances in the log is a key factor to discover high quality models.

3 Related Work

Reducing event logs to only significant behaviors is a common practice to improve model quality. Some variants of process discovery algorithms, e.g., the inductive miner [10], directly incorporate filters to remove infrequent behavior. In [11], infrequent traces are qualified as outliers and suggested to be filtered out. Independent to process discovery algorithms, filtering methods like [5,6,12] remove outlier behaviors in event logs. These works show possibilities of improvement when using reduced event logs as the input for process discovery algorithms.

Another way to reduce log variability that causes simpler models is to extract only a small set of traces. Authors in [8] and [9] present different trace selection methods that improve the performance of process discovery algorithms using random and biased sampling. These works are close to the present paper's method as the size of the reduced log is considerably smaller than the original one. We aim to select the most representative traces for process discovery.

As our prototype selection uses a clustering method, we recall that trace clustering has been used in process mining to get several sub-models according to clustered sub-logs [13,14,15,16]. The quality of the sub-models is better than a single process model discovered on the whole event log. However, getting several process models may be a barrier for decision-makers who need a single overview of each process.

Finally, in [2] a genetic process discovery algorithm is proposed that benefits from conformance artifacts. However, this method is time-consuming for large real event logs and it is impractical using normal hardware.

4 Preliminaries

In this paper, we focus on sequences of activities, also called traces, that are combined into event logs.

Definition 1 (Event Log). *Let $\mathcal{A}$ be a set of activities. An event log is a multiset of sequences over $\mathcal{A}$, i.e., $L \in \mathbb{B}(\mathcal{A}^*)$ that is a finite set of words. A word, i.e., a sequence of activities, in an event log is also called a* trace.

Fig. 1 shows an example of event log L. The occurrence of trace-variant $\langle a, b, c, e, g\rangle$ in this event log is four.

A sub-log L_1 of a log L is a set of traces such that $L_1 \subseteq L$. A trace clustering method aims to find disjoint sub-logs according to the similarity between traces.

Definition 2 (Trace Clustering). *Given a log L, a trace clustering $\xi(L, n)$ is a partitioning of L in a set of sub-logs $\{L_1, L_2 \ldots, L_n\}$ such that $\forall_{i \neq j} (L_i \cap L_j = \emptyset)$ and $\biguplus_{i=1:n} L_i = L$.*

We usually need a distance metric to cluster objects. One distance metric that is widely used to cluster words is Edit distance.

Definition 3 (Edit distance). *Let $\sigma, \sigma' \in \mathcal{A}^*$, edit distance function $\triangle(\sigma, \sigma') \rightarrow \mathbb{N}$ returns the minimum number of edits that are required to transform σ to σ'.*

We assume that an edit operation can only be a deletion or an insertion of an activity in a trace. To give an example, $\triangle(\langle a, c, f, e, d\rangle, \langle a, f, c, a, d\rangle) = 4$ corresponding to two deletions and two insertions.

Some clustering algorithms return a medoid for each cluster that is a representative object of that cluster. In this paper, we also return prototypes as medoids which have the closest distance with other objects in their cluster.

Definition 4 (Prototypes). *Let $\delta : \mathbb{B}(\mathcal{A}^*) \rightarrow \mathcal{A}^*$ be a function that for each sub-log L_i returns $p_i \in L_i$ which has the the minimum distance with other traces in that sub-log, i.e., $\sum_{\sigma \in L_i} (\triangle(p_i, \sigma))$. For a trace clustering $\xi(L, n) = \{L_1, L_2, \ldots, L_n\}$,* prototypes *are a set $P = \{p_i = \delta(L_i) : L_i \in \xi(L, n)\}$.*

In other words, a prototype is a unique trace-variant that represents a sub-log.

A *process model*, commonly Petri net or BPMN, describes a set of traces. As the present paper does not propose a specific notation for process models, we define a process model by its describing behavior.

Definition 5 (Runs of Process Model). *Let M be a process model with a set of activities $\mathcal{A}$. We define a set of all possible traces that can be executed by M as $Runs(M) \subseteq \mathcal{A}^*$. In case of having loop in the model, this set is infinite.*

For example, Fig. 1a describes six traces; therefore, we have $Runs(M) = \{\langle a, c, b, e, g\rangle, \langle a, b, c, e, g\rangle, \langle a, d, e, g\rangle, \langle a, c, b, e, f\rangle, \langle a, b, c, e, f\rangle, \langle a, d, e, f\rangle\}$.

A process model and an event log may have some deviations. For instance, $\langle a, f\rangle$ is not in the described behavior of the model that is presented in Fig. 1a). It is shown in [17] that using the following formula, we can measure the fitness of a model and a traces based on the edit distance function.

$$trace\_fitness(\sigma_L, M) = 1 - \frac{\min\limits_{\sigma \in Runs(M)} \triangle(\sigma_L, \sigma)}{|\sigma_L| + \min\limits_{\sigma \in Runs(M)} |\sigma|} \tag{1}$$

The fitness of an event log and a model is a weighted average of the *trace-fitness* of trace logs. Thus, log traces with a higher frequency have higher weights.

In contrast, precision shows how much behavior in a model exists in occurs in the log. In this paper, we refer by $Precision(L, M)$ to ETC [18] as it has a high performance computation; however any other precision metrics can be used. To balance between the two main metrics, we use the F-Measure [19]:

$$F\text{-}Measure = 2 \times \frac{Precision \times Fitness}{Precision + Fitness} \tag{2}$$

5 Incremental Prototype Selection for Process Discovery

In this section, we explain the details of our approach to use the selected *Prototypes*, i.e., a subset of traces, for representing the entire log as a process model. As explained, we use a clustering approach to select the representative prototypes for process discovery. The schematic view of the proposed method is presented in Fig. 2. The method contains the following four main steps:

1. *Clustering for prototype selection*: to select prototypes using a clustering method.
2. *Model discovery*: discovering a model based on the selected prototypes.

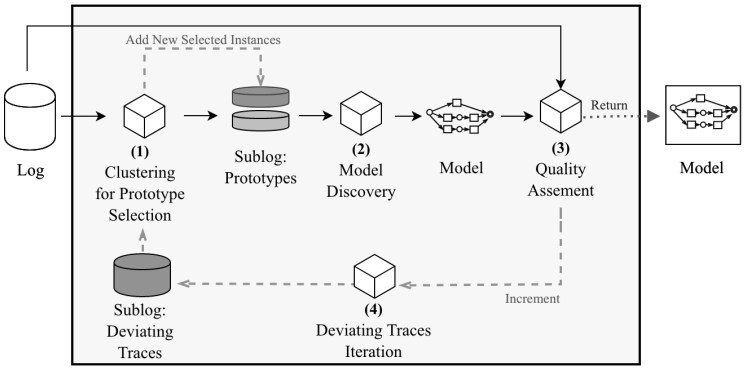


Fig. 2: Structure of the Prototype Selection Approach

3. *Quality assessment*: to evaluate the discovered model based on the original event log, conformance artifacts are computed.
4. *Iteration over deviating traces*: while quality metrics improve, we iterate the method (from Step 1) on the deviating traces of the last discovered model.

The prototype selection is an iterative process; then, the sub-log of selected prototypes gently grows in each iteration. During an iterative process, we expect that fitness increases while the precision value decreases. Here, we explain each step in more detail.

1- Clustering for Prototype Selection By applying process discovery algorithms directly on a complete event logs, we usually obtain complex and imprecise process models. As presented in [20,8] by modifying and sampling event logs we are able to improve results of process discovery algorithms in terms of F-measure. We apply clustering to extract a very small set of representatives traces, i.e., prototypes. In this regards, we use K-medoids [21] to cluster traces in K sub-logs by considering their similarity (using the Edit distance function). This algorithm works as follows:

1. Select randomly K (i.e., the number of clusters) traces in L as medoids.
2. Create, or update, clusters by associating each trace to its closest medoid based on the Edit distance metric.
3. For each cluster, redefine the medoid as the prototype of the cluster according to Definition 4 using δ function. If medoids haven't changed, return the K prototypes. Otherwise, do Step 2 again.

The prototypes are then added to the set of *selected prototypes* which is empty at the first iteration of our method (see Fig. 2). For example, for the event log that is presented in Fig. 1e, applying the clustering with K=3 in the first iteration gives $\langle a, b, c, g\rangle$, $\langle a, c, b, e, f\rangle$ and $\langle a, d, e, f\rangle$ as prototypes.

2- Model Discovery After selecting a set of prototypes, we discover a descriptive view of it, i.e., a process model. In this regard, we are flexible to use any process discovery algorithm. However, it is recommended to use methods that guarantee to return sound process models as it is necessary for fitness computation. By discovering a process model from the selected prototypes, we will have a general view of what is going on in the process and position different log traces w.r.t, this model.

3- Quality Assessment To ensure that the discovered process model via prototypes conforms to the whole event log, we incorporate quality assessment evaluations in our method. We use the F-measure to get a good balance between fitness and precision. The metric is computed by considering the original event log and the process model created based on the selected prototypes.

4- Iteration over Deviating Traces The F-measure is computed for the first time after the initialization step that selects a first set of prototypes. Thereafter, the proposed method starts an iterative procedure. In each iteration, the method first finds the deviating traces that are formally defined as follows.

Table 1: Some information of the real-life event logs used in the experiments.

Event Log	Activities#	Traces#	Variants#	DF Relations#
BPIC_2012[23]	23	13087	4336	138
BPIC_2018_Insp.[24]	15	5485	3190	67
BPIC_2019[25]	44	251734	11973	538
Hospital[26]	18	100000	1020	143
Road [27]	11	150370	231	70
Sepsis[28]	16	1050	846	115

Definition 6 (Deviating Traces). *Let M be a process model and L be an event log. The* Deviating Traces *is* $L_d=\{\sigma\in L : \sigma\notin Runs(M)\}$.

After finding the deviating traces, we look for other representative traces among them like what we did in Step 1, i.e., using clustering. Then, the new prototypes will be added to the previous ones (see Fig. 2). Here, for clustering of deviating traces, we are able to use similar or different K compared to the first step. Thereafter, we apply again the process discovery algorithm to find a new process model and so on. Afterward, we compare the previous and current F-measure values. The iterative procedure stops when the quality of the new discovered process model is lower than the previous one. By increasing in the number of the selected prototypes, we expected that the fitness of discovered model is increased, but its precision is decreased. So we use F-measure that balances the metrics. We make the hypothesis that process discovery algorithms tend to approach high fitness and adding traces in the input raises the fitness of the whole log and decreases the precision. This hypothesis is commonly true (as also assumed in [22]). Therefore, the algorithms stops when there is no improvement in F-measure of the discovered process model of prototypes.

6 Experiments

In this section, we investigate whether the proposed method results in process models with higher quality. To apply the proposed method, we implemented the *Prototype Selection* plug-in in `ProM` framework.The plug-in takes an event log as input and outputs the discovered model and the selected prototypes. As presented above, our method uses two parameters, i.e., the number of clusters/prototypes that will be selected in each iteration and the process discovery algorithm. To simplify the plug-in, we consider the same cluster size for both Step 1 and 4. We ported the *Prototype Selection* plug-in into `RapidProM` that allows us to apply the proposed method on various event logs with different parameters. `RapidProM` is an extension of `RapidMiner` that combines scientific workflows with a range of (`ProM`-based) process mining algorithms.

The experiments have been conducted on six real event logs of different fields, from healthcare to insurance. Event logs have different characteristics which are given in Tab. 1.

In the following, we first position the proposed method compared to some state-of-the-art preprocessing methods. Later, we analyze the clustering method for selecting prototypes.

Table 2: Average of precision, fitness, and F-Measure for different methods.

Miner	Log	Nothing			Prototype Selection			Sampling			Statistical		
		Fitness	Precision	F-Measure	Fitness	Precision	F-Measure	Fitness	Precision	F-Measure	Fitness	Precision	F-Measure
ILP	BPIC-2012	1,00	0,12	0,21	0,75	0,74	0,65	0,88	0,24	0,37	1,00	0,12	0,22
	BPIC-2018-Ins.	1,00	0,13	0,22	0,88	0,64	0,68	0,96	0,37	0,51	1,00	0,16	0,28
	BPIC-2019	1,00	0,36	0,53	0,89	0,84	0,82	0,98	0,6	0,73	1,00	0,52	0,68
	Hospital	1,00	0,39	0,57	0,87	0,86	0,84	0,98	0,59	0,72	1,00	0,44	0,61
	Road	1,00	0,53	0,69	0,86	0,89	0,85	0,91	0,8	0,84	1,00	0,61	0,76
	Sepsis	1,00	0,2	0,34	0,81	0,68	0,65	0,94	0,39	0,53	1,00	0,22	0,35
IMi	BPIC-2012	0,83	0,59	0,67	0,78	0,78	0,76	0,89	0,56	0,68	1,00	0,54	0,7
	BPIC-2018-Ins.	0,97	0,52	0,67	0,59	0,79	0,66	0,8	0,62	0,65	0,94	0,53	0,66
	BPIC-2019	0,97	0,48	0,64	0,78	0,75	0,74	0,95	0,68	0,76	1,00	0,83	0,9
	Hospital	0,83	0,86	0,83	0,85	0,93	0,88	0,96	0,86	0,9	1,00	0,84	0,91
	Road	0,88	0,65	0,74	0,88	0,9	0,88	0,9	0,95	0,92	1,00	0,82	0,9
	Sepsis	0,9	0,56	0,65	0,84	0,71	0,75	0,91	0,56	0,67	1,00	0,49	0,66
SM	BPIC-2012	0,89	0,72	0,79	0,76	0,89	0,81	0,84	0,69	0,76	0,99	0,59	0,74
	BPIC-2018-Ins.	0,9	0,71	0,79	0,87	0,83	0,83	0,88	0,76	0,81	0,92	0,67	0,77
	BPIC-2019	0,98	0,76	0,85	0,85	0,97	0,9	0,96	0,66	0,76	1,00	0,44	0,61
	Hospital	0,99	0,9	0,94	0,9	1,00	0,94	0,92	0,98	0,94	1,00	0,97	0,98
	Road	0,89	0,9	0,89	0,89	1,00	0,94	0,87	0,99	0,93	1,00	0,95	0,98
	Sepsis	0,87	0,62	0,71	0,85	0,75	0,77	0,86	0,71	0,76	0,99	0,43	0,6

6.1 Process Discovery Improvement

Here we aim to find out how the proposed method is able to improve the results of different process discovery algorithms. As the *Prototype Selection* has two parameters, i.e., the number of clusters and the discovery algorithm, we show results over a set of different settings. We repeated the experiments for 2 to 9 clusters and we used the inductive miner [29], the ILP miner [30], and the split miner [31]. Moreover, we compared our work with two trace sampling methods [8,9], i.e., referred to *Sampling* and *Statistical* respectively. For both of these methods, we ran the experiments with 20 different settings. When we use the preprocessing methods, we used the inductive miner with its filtering mechanism set to 0 and the default setting for the split miner. We also compared our method to normal process discovery algorithms, i.e, discovery without preprocessing which we denote it by *Nothing* in the experiments. For this case, we ran a set of experiments with 50 different settings for the inductive miner (IMi) and 100 for the split miner. For the ILP miner, we just run the experiment without its internal filtering mechanism.

Tables 2 and 3 show the average results of the experiments over the different settings. It is shown in Tab. 2 that for most of the cases, the F-Measure of discovered process models using the prototype selection method is higher than other preprocessing techniques and generally, the proposed method leads to provide more precise process models.

For simplicity, in Tab.3, we consider two metrics that measure the complexity of discovered process models. *Model Size* of process models is a combination of the number of transitions, places, and arcs that connected them. Another metric is the Cardoso metric [32] that measures the complexity of a process

Table 3: Comparison of simplicity measures for different preprocessing methods.

Miner	Log	Nothing		Prototype Selection		Sampling		Statistical	
		Cardoso	Model Size	Cardoso	Model Size	Cardoso	Model Size	Cardoso	Model Size
ILP	BPIC-2012	163	33×28×426	83,7	25.7×23.3×172.7	182,5	31×26.6×449.5	234	33.5×27.7×593.7
	BPIC-2018-Ins.	142	26×19×324	123	25.3×17×246	110,2	24×18.1×238.3	112,7	24.3×18.3×234.7
	BPIC-2019	561	61×46×1550	45,7	16×13.3×90.7	379	45.6×39.6×1157.2	365	47.3×39.5×1064.3
	Hospital	120	29×22×440	16,7	10.3×13×38	106,2	22.5×19.8×370.1	103,8	21.5×20.3×386.7
	Road	67	16×15×150	18,5	10.5×12×39.3	52,5	16.4×14.8×115.1	53,2	17×15×109.3
	Sepsis	209	31×20×470	73	19×17×153.3	126,9	25.7×18.7×287.6	187	28.3×20×407.7
IMi	BPIC-2012	27	21×44×90	45,7	34.7×45.7×99.3	47,1	34.7×52.3×111.9	45,3	32.8×55.3×115.7
	BPIC-2018-Ins.	25	19×33×68	16,3	14×23.3×49.3	13,3	11.9×22.8×47.7	13,5	12×23.2×48.3
	BPIC-2019	32	21×65×132	20	15.3×22.3×45.3	34,6	24.5×60×123	38,2	26.2×62.2×128.3
	Hospital	43	31×52×108	24,7	19×26×54.7	33	23×41.5×84.2	35	23.5×43.8×88
	Road	19	19×25×56	10,5	10×15.5×31	18,3	15.7×24.6×51	21,8	18.7×28.2×58.7
	Sepsis	22	17×33×68	17	15×23.3×48.7	27,5	22.4×33.4×72.7	21,7	17.5×32.5×67.7
SM	BPIC-2012	106,8	65.9×101×240.2	41	31×42.3×86	85,8	55×84×179.8	102,7	62.2×98.5×218.8
	BPIC-2018-Ins.	56,5	33.2×56×115.4	39,8	24.8×39.8×80.3	42,6	25.8×43.5×87.3	44,7	26×45.7×91.3
	BPIC-2019	393,3	139×388.1×894.3	19,7	15.7×22×44	255,4	98.5×259.5×526.2	210	89.8×215×430.3
	Hospital	135,7	62.5×137.2×274.3	15	13×16×32	77,7	42.8×79.4×158.8	75,3	42.7×76.8×153.7
	Road	46,2	30.6×41.9×108.7	13,7	12.7×13.7×30	29,8	23×28.6×69.3	29,8	21.5×29.8×68.7
	Sepsis	97,3	49.5×94.5×232	30,8	23.2×32.8×65.7	68,3	34.9×70.6×145.8	95,7	41.2×99×198

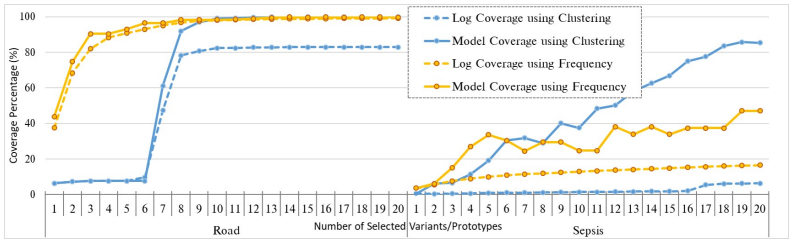
model by its complex structures, i.e., *Xor*, *Or*, and *And* components. For both of these measures, a lower value means less complexity and consequently a simpler process model. Results show that we can have much simpler process models using the proposed method. By considering both tables, we see that the presented method helps to get more precise and simpler models in most of the event logs.

6.2 Using Clustering for Prototype Selection

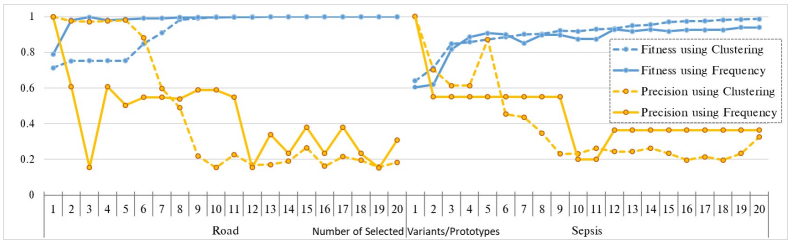
Here, we aim to find how by selecting prototypes using the clustering method we improve the quality of process models. We increased the number of prototypes from 1 to 20 and analyze the quality of the resulted models for Sepsis [28], and Road [27] event logs. We used the inductive miner without the internal filtering for model discovery. In Fig. 3, we compared the results of prototype selection based on clustering and the most frequent variants on the discovered models.

In Fig. 3a, the log coverage shows how many percentage of the traces in the event log, is corresponds to the selected prototypes. Moreover, the model coverages indicates that how many percentages of traces in the event log is replayable (or perfectly fitted) by the discovered process model. For example, in the Sepsis event log, by selecting eight prototypes, i.e., corresponds to 5% of traces, the discovered process model is able to perfectly replay 35% of the traces in the event log. Fig. 3a shows that process discovery algorithms depict much behavior in the process model compared to the given event log. For event log with high frequent traces, e.g., *Road*, when we select very few process instances, by selection based on frequency, we usually have higher model coverage. However, when we select more than 10 prototypes, or for event logs with lots of unique variants, e.g., *Sepsis*, the model coverage of clustering method is higher.

In Fig. 3b, we see how by increasing the number of prototypes, generally fitness increases and precision decreases. This reduction is higher when we select based on frequency. Results show that we can discover a high fitted process



(a) Coverage analysis



(b) Fitness and precision analysis

Fig. 3: Effects of increasing the number of selected prototypes on the quality issues of discovered process models using frequency and clustering methods.

model without giving just a few prototypes to process discovery algorithms. This experiment shows that using the clustering algorithm we can choose the more representative prototypes specifically if the log has lots of unique behavior.

7 Conclusion

In this paper, we proposed an incremental method to select prototypes of the event logs in order to generate simple and precise process models having a good F-measure. It clusters the traces in the event log based on their control-flow distances. Afterward, it returns the most representative instance for each cluster, i.e., the prototype. We discover a process model of the selected prototypes which is analyzed by common conformance metrics. Then, the method recessively selects new prototypes from deviating traces. A novel set of traces is added to the process discovery algorithm which improves fitness while decreasing precision.

To evaluate the proposed method, we have developed it in `ProM` and `RapidProM`, and have applied the proposed prototype selection method on six real event logs. We compared it with other state-of-the-art sampling methods using different process discovery algorithms. The results indicate that the proposed method is able to select process instances properly and help process discovery algorithms to return process models with a better balance between quality measures. Discovered models are less complex and, consequently, easier to understand. Another advantage of our method is that it is more stable in chosen settings of parameters and tends to return process models with higher quality.

As future work, it is possible to find prototypes using more advanced clustering methods and measures that are proposed in the literature. Indeed, the

weakness of the F-measure is the fact that it is an average of fitness and precision, which blurs the understanding of the chosen model. Instead, the use of F_β-measure introduced in [33] can help one to balance between the two criteria. Moreover, we aim to use prototypes for other purposes, e.g., conformance checking and performance analysis. One limitation of our method is it may find a local optimum rather than the global optimum. We plan to use an adjustable number of clusters for both initiating phase and incremental steps.

Acknowledgment

We thank Prof. Josep Carmona, Dr. Thomas Chatain and Dr. Sebastiaan J. van Zelst for comments that greatly improved the work. We also thank the Alexander von Humboldt (AvH) stiftung for supporting this research.

References

1. van der Aalst, W.M.P.: Process Mining - Data Science in Action, Second Edition. Springer Berlin Heidelberg (2016)
2. Buijs, J.C., van Dongen, B., van der Aalst, W.M.P.: On the Role of Fitness, Precision, Generalization and Simplicity in Process Discovery. In: CoopIS 2012, Rome, Italy. (2012) 305–322
3. Carmona, J., van Dongen, B., Solti, A., Weidlich, M.: Conformance Checking. Springer (2018)
4. Bose, R.J.C., Mans, R.S., van der Aalst, W.M.P.: Wanna improve process mining results? In: IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2013, Singapore, 16-19 April, 2013. (2013) 127–134
5. Conforti, R., Rosa, M.L., ter Hofstede, A.H.M.: Filtering out infrequent behavior from business process event logs. IEEE Trans. Knowl. Data Eng. **29**(2) (2017) 300–314
6. Fani Sani, M., van Zelst, S.J., van der Aalst, W.M.P.: Improving process discovery results by filtering outliers using conditional behavioural probabilities. In: Business Process Management Workshops, Revised Papers. (2017)
7. van der Aalst, W.M.P., Bolt, A., van Zelst, S.: Rapidprom: Mine your processes and not just your data. CoRR **abs/1703.03740** (2017)
8. Fani Sani, M., van Zelst, S.J., van der Aalst, W.M.P.: The impact of event log subset selection on the performance of process discovery algorithms. In: New Trends in Databases and Information Systems, Slovenia. (2019) 391–404
9. Bauer, M., Senderovich, A., Gal, A., Grunske, L., Weidlich, M.: How much event data is enough? a statistical framework for process discovery. In: CAiSE 2018, Tallinn, Estonia, 2018, Proceedings. (2018) 239–256
10. Leemans, S.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs containing infrequent behaviour. In: BPM 2013 International Workshops, Beijing, China. (2013) 66–78
11. Ghionna, L., Greco, G., Guzzo, A., Pontieri, L.: Outlier detection techniques for process mining applications. In: Foundations of Intelligent Systems, 17th International Symposium, ISMIS. (2008) 150–159
12. Fani Sani, M., van Zelst, S.J., van der Aalst, W.M.P.: Applying sequence mining for outlier detection in process mining. In: CoopIS 2018, Malta. (2018) 98–116

13. Tax, N., Sidorova, N., Haakma, R., van der Aalst, W.M.P.: Mining local process models. J. Innov. Digit. Ecosyst. **3**(2) (2016) 183–196
14. Boltenhagen, M., Chatain, T., Carmona, J.: Generalized alignment-based trace clustering of process behavior. In: Application and Theory of Petri Nets and Concurrency - 40th International Conference, Germany. (2019) 237–257
15. Weerdt, J.D., vanden Broucke, S.K.L.M., Vanthienen, J., Baesens, B.: Leveraging process discovery with trace clustering and text mining for intelligent analysis of incident management processes. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC. (2012) 1–8
16. De Weerdt, J., Vanden Broucke, S., Vanthienen, J., Baesens, B.: Active trace clustering for improved process discovery. IEEE Trans. Knowl. Data Eng. **25**(12) (2013) 2708–2720
17. Fani Sani, M., van Zelst, S.J., van der Aalst, W.M.P.: Conformance checking approximation using subset selection and edit distance. In: CAiSE 2020, Grenoble, France, June 8-12, 2020, Proceedings. (2020) 234–251
18. Muñoz-Gama, J., Carmona, J.: A fresh look at precision in process conformance. In: Business Process Management - 8th International Conference, BPM 2010, Hoboken, NJ, USA, September 13-16, 2010. Proceedings. (2010) 211–226
19. Van Rijsbergen, C.J.: Information retrieval. (1979)
20. Fani Sani, M., van Zelst, S.J., van der Aalst, W.M.P.: Repairing outlier behaviour in event logs using contextual behaviour. Enterp. Model. Inf. Syst. Archit. Int. J. Concept. Model. **14** (2018) 5:1–5:24
21. De Amorim, R.C., Zampieri, M.: Effective spell checking methods using clustering algorithms. In: Recent Advances in Natural Language Processing, RANLP 2013, 9-11 September, 2013, Hissar, Bulgaria. (2013) 172–178
22. Augusto, A., Dumas, M., La Rosa, M.: Metaheuristic optimization for automated business process discovery. In: Business Process Management - 17th International Conference, Vienna, Austria. (2019) 268–285
23. van Dongen, B.F.: BPI Challenge 2012. Eindhoven University of Technology. Dataset. (2012)
24. van Dongen, B.F., Borchert, F. (Florian): BPI Challenge 2018 Eindhoven University of Technology. Dataset. (2018)
25. van Dongen, B.F.: BPI Challenge 2019. Eindhoven University of Technology. Dataset. (2019)
26. Mannhardt, F.: Hospital Billing-Event Log. Eindhoven University of Technology. Dataset. Eindhoven University of Technology. Dataset (2017) 326–347
27. De Leoni, M., Mannhardt, F.: Road traffic fine management process. Eindhoven University of Technology. Dataset (2015)
28. Mannhardt, F.: Sepsis cases-event log. Eindhoven University of Technology (2016)
29. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs containing infrequent behaviour. In: Business Process Management Workshops - BPM 2013 International Workshops. (2013) 66–78
30. van Zelst, S.J., van Dongen, B.F., van der Aalst, W.M.P., Verbeek, H.M.W.: Discovering relaxed sound workflow nets using integer linear programming. CoRR (2017)
31. Augusto, A., Conforti, R., Dumas, M., Rosa, M.L., Polyvyanyy, A.: Split miner: automated discovery of accurate and simple business process models from event logs. Knowl. Inf. Syst. **59**(2) (2019) 251–284
32. Lassen, K.B., van der Aalst, W.M.P.: Complexity metrics for workflow nets. Inf. Softw. Technol. **51**(3) (2009) 610–626
33. Chinchor, N.: Muc-4 evaluation metrics, ACL (1992)