Determining the Number of Trace Clusters: a Stability-based Approach

Pieter De Koninck and Jochen De Weerdt

KU Leuven
Research Centre for Management Informatics
Faculty of Economics and Business
Naamsestraat 69, B-3000 Leuven, Belgium
pieter.dekoninck@kuleuven.be
jochen.deweerdt@kuleuven.be

Abstract. Given the complexity of real-life event logs, several trace clustering techniques have been proposed to partition an event log into subsets with a lower degree of variation. In general, these techniques assume that the number of clusters is known in advance. However, this will rarely be the case in practice. Therefore, this paper is the first to present an approach to determine the appropriate number of clusters in a trace clustering context. In order to fulfil this objective, a stability-based method for identifying the most appropriate number of trace clusters is proposed. The method involves the design of tailored resampling strategies and cluster similarity metrics. Regarding practical validation, our approach is tested on multiple real-life datasets to investigate the workings of the different components. Our results suggest that our method is successful in identifying the right number of trace clusters.

Keywords: stability, trace clustering, validity, log perturbation

1 Introduction

Trace clustering is the partitioning of process instances into different groups, called trace clusters, based on their similarity. A wide variety of trace clustering techniques have been proposed, differentiated by their clustering methods and biases. The driving force behind these proposed techniques is the observation that real-life event logs are often quite complex and contain a large degree of variation. Since these event logs are often the basis for further analysis like process model discovery or compliance checking [25], partitioning dissimilar process instances into separate trace clusters is deemed appropriate. Although a wide array of techniques has been proposed, none of them makes any assertions on the correct number of clusters. Therefore, this paper is the first to propose a suitable approach for determining the most plausible number of clusters. Since our approach can be applied to any trace clustering technique, it raises the applicability of trace clustering techniques in general, and the validity of their trace clustering solutions.

Our approach is based on the stability of trace clustering solutions. Intuitively, it can be expected that trace clustering solutions are more stable at the correct number of clusters. Therefore, we develop a general framework to assess the stability of trace clustering solutions. When repeatedly applied to an event log for a range of potential number of clusters, one can compare the stability scores obtained for each number of clusters. The result with the highest stability can be considered the most appropriate number. A number of elements are conceived to construct our approach: specifically, two approaches are proposed to resample event logs. Likewise, two methods are provided for calculating the similarity of clustering solutions. Finally, the concept of normalization and a calculation strategy are supplied. Each of these elements is thoroughly evaluated on four real-life event logs, resulting in the conclusion that the stability-based framework configured with model-based similarity metrics and a noise induction-based resampling strategy can lead to the correct identification of the appropriate number of clusters[1].

2 Determining the number of clusters

In traditional clustering, numerous approaches have been suggested for assessing the adequate number of clusters. A taxonomy of approaches for determining the number of clusters has been presented in [22]. The most straightforward approach is to incorporate domain knowledge, either by directly adjusting your algorithm to suit the knowledge of a domain expert or by post-processing the results to adhere to this knowledge. In general, however, it is unlikely that such domain knowledge exists and is available for an event log. Creating an approach based on the specific generation of trace clusters will not be applicable for each existing trace clustering technique either. Therefore, we propose to adapt approaches based on the post-processing of partitions. According to the taxonomy of [22], possible post-processing approaches can be based on variance, structure, consensus and resampling. The most commonly known variance-based method is probably the gap statistic [24], which is based on the within-cluster sum of squares using Euclidean distance. Likewise, structural approaches use indices to compare within-cluster cohesion to between-cluster separation [22]. It is clear that one would prefer a number of clusters where the within-cluster cohesion and the between-cluster separation are both large. As a third group of approaches, consensus clustering refers to choosing the number of clusters based on the agreement between different cluster solutions. These different solutions can be obtained by applying different clustering techniques, by applying the same clustering technique to perturbed versions of the same data set, or by randomly resetting initial centroids (in a centroid-based technique). Intuitively, the consensus between different clustering solutions should be higher at the true number of clusters. The final group of post-processing approaches is based on resampling, and is related to consensus clustering in its intuition: a number of

[1] Our approach is implemented as an experimental ProM-plugin which can be found on *http://www.processmining.be/clusterstability/*.

iterations are performed in which sub-sampled, bootstrapped or noisy versions of the original data set are clustered. The resulting partitions are then expected to be more similar at the appropriate number of clusters.

With regards to applicability for trace clustering, adapting variance- or structure-based approaches to trace clustering might not be straightforward, since a distance measure is needed. To calculate distances between traces, features would have to be derived from these traces. Considering that certain trace clustering techniques deliberately avoid 'featurizing' traces [6], this not deemed an appropriate route for trace clustering. Consensus- and resampling-based approaches generally do not enforce direct distances between traces, therefore our approach will further draw on these methods in the upcoming sections.

3 Stability of Trace Clustering

The approach proposed in this paper is a resampling-based approach, inspired by a methodology for stability-based validation of clustering solutions in [18], which was adapted for biclustering solutions in [20]. In [18], it was shown to be an effective method for discovering the appropriate number of clusters on simulated and gene expression data.

In [18,20], resampling/perturbation strategies, learning algorithms, and solution similarity metrics are proposed that are specifically designed for general (bi)clustering problems. The general intuition is that clustering solutions should remain more stable at the true number of clusters that at others. As such, this paper contributes by proposing a stability-based approach for determining the correct number of trace clusters. Our approach leverages the so-called "log perturbation stability", which is the adaptation of general resampling to the process mining domain. In Figure 1, our general stability approach is depicted. Tailoring the framework to trace clustering entails the configuration of three main components, i.e. the perturbation strategy (step 1), the solution similarity computation (step 3), a stability index calculation (step 4). In addition, a trace clustering technique should be chosen (step 2). This stability is then normalized with respect to the stability of a random clustering on the same perturbed event logs (step 5).

The steps of our approach thus become:

1. **Step 1**: Given an event log L, and a log perturbation function $P()$, create n perturbed versions of the event log: $P_1(L)$ to $P_n(L)$.
2. **Step 2**: Create a clustered log CL by applying a trace clustering technique $TC()$ to the original event log: $CL = TC(L)$ and to the perturbed event logs: $CL_i = TC(P_i(L))$ with $i \in \{1..n\}$.
3. **Step 3**: Given a similarity index $I(CL_x, CL_y)$, quantify the similarity between the clustering of the original dataset and the clustering of the perturbed dataset as $I(CL, CL_i)$.

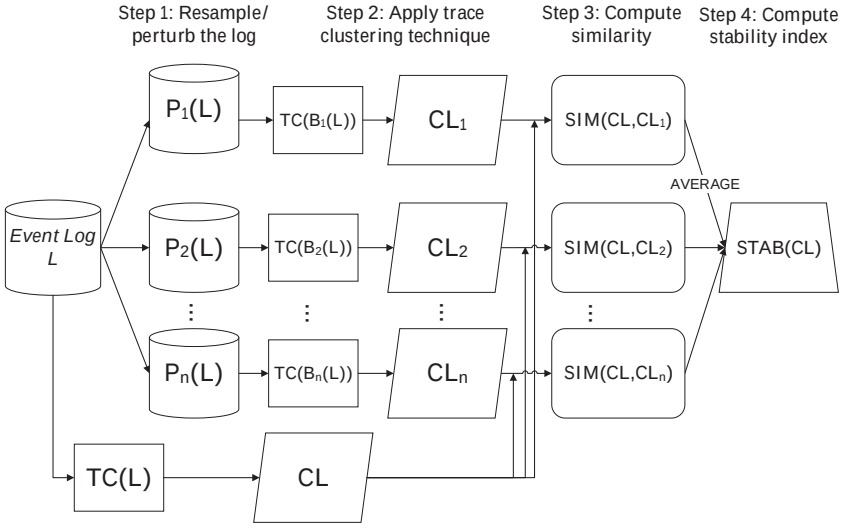

Fig. 1. A visualization of the proposed approach for calculating the stability of a clustered event log, based on a similar diagram in [20]. A normalized version of this stability is calculated at different numbers of clusters to determine the optimal number of clusters.

4. **Step 4**: Average these similarity measures to create a stability metric for event log L and trace clustering technique $TC()$ as

$$S^{TC} = \frac{1}{n} \sum_{i=1}^{n} I(TC, TC_i) \tag{1}$$

5. **Step 5**: Normalize with respect to the stability of a random clustering technique S^R over the same set of perturbed event logs:

$$\bar{S}^{TC} = \frac{S^{TC} - S^R}{1 - S^R} \tag{2}$$

Observe that a higher value for $\bar{S}^{TC}$ indicates a better stability of the solution. This metric should be evaluated at different numbers of clusters, at which point the best scoring number of clusters should be chosen. In the remainder of this section, we describe the three main components of our approach: possible perturbation strategies based on resampling and noise induction (Section 3.1), computation of solution similarity based on mutual information or process model similarity metrics (Section 3.3), calculation of the stability index based on a window-based approach (Section 3.4), and normalization of the stability with respect to random stability (Section 3.5).

3.1 Step 1: Log perturbation strategy

Perturbing event logs essentially boils down to three options: either some behaviour is removed, or some behaviour is added, or a combination of both. There are many different ways to do this, as argued in [22]: sub-sampling, data-splitting, bootstrapping and noise induction. Regarding the removal of behaviour, event log perturbation can be approached through case-level resampling in a random fashion, which is closely related to classical bootstrapping [3]. Note that case-level bootstrapping an event log becomes trace-level bootstrapping. When dealing with event logs, an important consideration is whether to bootstrap process instances or distinct process instances (i.e. the effect of imbalance on the bootstrap sample). An alternative to random resampling is systematic leave-one-out cross-validation, which can be considered a form of 'data-splitting'.

Finally, regarding the addition of behaviour, slightly perturbing event logs strongly relates to the concept of adding noise to the log. In [21], four types of noise were initially defined: remove head, remove tail, remove body, and swap tasks. In [4], the removal of a single task was added as a noise induction scheme, together with the combination of all previous noise types. These noise induction types have already been used to evaluate robustness of process discovery techniques, for instance in [8],[15].

Taking these aspects into consideration, the log perturbation strategy underlying our stability assessment framework is as follows. First, behaviour can be removed through a resampling procedure, which is essentially sub-sampling at the level of distinct process instances. However, to make the resampling a bit less naive, the probability that a distinct process instance is removed, is inversely proportional to the frequency with which this distinct process instance is present in the event log. Secondly, behaviour can be added through noise induction. Though several noise types were proposed in [21], we opt to include three types of noise: removing a single event, swapping two events, and adding a random single event (from the log activity alphabet) at a random place in the process instance. Noise induction is performed at the process instance level. For both removal of behaviour (sub-sampling at the distinct process instance level) and addition of behaviour (noise induction at the process instance level), a percentage of affected instances should be chosen.

3.2 Step 2: Trace clustering technique

In the next step, a certain trace clustering algorithm is applied. The choice of a suitable algorithm is up to the analyst, and a thorough overview of the existing techniques is beyond the scope of this paper. Nonetheless, it is noteworthy that two broad categories of trace clustering techniques exist: those that map traces onto a vector space model or quantify the similarity between two traces [1],[2],[7],[12],[13],[16],[23]; and those that take the quality of the underlying process models into account [6],[11].

3.3 Step 3: Solution similarity computation

In this section, two distinct approaches for computing the similarity between two clusterings will be described. One is inspired by information metrics from the consensus clustering domain, and one is inspired by similarity metrics from the process modelling domain.

On the one hand, we propose a consensus clustering-based metric. It is called the Normalized Mutual Information (NMI), and was proposed by [14]. It is a measure for the extent to which two clusterings contain the same information. Here, this mutual information is conceptually defined as the extent in which two process instances are clustered toghether in both clusterings. Let k_a be the number of clusters in clustering a, k_b the number of clusters in clustering b, n the total number of traces, n_i^a the number of elements in cluster i in clustering a, n_j^b, the number of elements in cluster j in clustering b, and n_{ij}^{ab} the number of elements present in both cluster i in clustering a and cluster j in clustering b. The NMI is then defined as:

$$I_{NMI}(a,b) = -2\frac{\sum_{i\in a}^{k_a}\sum_{j\in b}^{k_b} n_{ij}^{ab} log(\frac{n_{ij}^{ab} n}{n_i^a n_j^b})}{\sum_{i\in a}^{k_a} n_i^a log(\frac{n_i^a}{n}) + \sum_{j\in b}^{k_b} n_j^b log(\frac{n_j^b}{n})} \tag{3}$$

On the other hand, we propose a metric based on the similarity between discovered process models. Rather than measure the similarity by counting the number of elements that are included in the same cluster in both cluster solutions (i.e. measuring the consensus between both clusterings), each different cluster is used to discover a process model. Then, a process model similarity metric is used to measure the similarity between these discovered process models. This is represented conceptually in Figure 2.

A plethora of process discovery techniques and process similarity metrics exist that could be leveraged for this purpose. With regards to process discovery techniques, an efficient and robust technique is preferred. Therefore, we propose the usage of Heuristics miner [27]. It mines a heuristic net, which is converted to a Petri net. With regards to process model similarity, our preference goes out to the structural graph-edit distance (GED) similarity metric [9], though behavioural metrics such as causal footprints [10] or behavioural profiles [26] could be used as well. Finally, our similarity metric for trace clustering solutions is summarized in Equation 4, where k_a is the number of clusters in clustering a, n_i is the number of elements in cluster i of clustering a, and $sim_{HG}(i,j)$ is the graph-edit distance similarity between the converted heuristic net mined from cluster i of clustering a and the converted heuristic net mined from cluster j of clustering b.

$$I_{HG}(a,b) = \frac{\sum_{i\in a}^{k_a} n_i \max_{j\in b}(sim_{HG}(i,j))}{\sum_{i\in a}^{k_a} n_i} \tag{4}$$

In [19], it is stated that a high-quality similarity index should have two characteristics: (1) it should take differences in cluster sizes into account, and (2) it

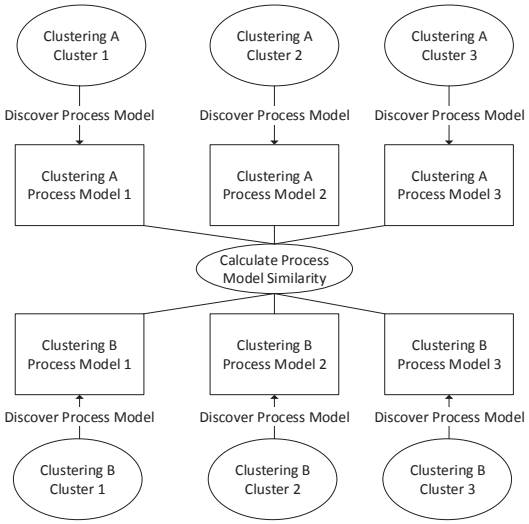

Fig. 2. Conceptual representation of the process model-based similarity metric, when two clusterings of three clusters each are under comparison.

should be symmetric. Note from Equation 3 that these properties are fulfilled for $I_{NMI}()$. Likewise, from Equation 4, it is clear that $I_{HG}(a, b)$ is weighted for the effects of different cluster sizes. However, it is not symmetric yet, i.e. $I_{HG}(a, b) \neq I_{HG}(b, a)$ due to the combination of weights and the max-operator. Therefore, we propose a final symmetric variant $\bar{I}_{HG}$:

$$\bar{I}_{HG}(a, b) = \frac{I_{HG}(a, b) + I_{HG}(b, a)}{2} \tag{5}$$

3.4 Step 4: Stability index computation

Next, in step 4 of our framework, the stability index is computed as an average over a number of iterations, as detailed in Algorithm 1 in the 'Stability'-function. Hereto, three extra input parameters are necessary: a minimal number of iterations r_{min}, a review window Δr and a maximal stability error ϵ_S. Typical values for these parameters are 20, 10, and 0.005 respectively. This iterative approach serves a double purpose: on the one hand, it ensures that the final stability is robust and sufficiently precise; on the other hand, it prevents unnecessary computation.

Algorithm 1 Stability evaluation

Input: L := Event log, TC := Trace clustering algorithm, P:= Perturbation strategy, I_s := similarity metric, k_{max}:= maximum number of clusters;
Input: $r_{min} := 20$, $\Delta r := 10$, $\epsilon_S := 0.005$; % Configuration
Output: k := number of clusters for which the normalized stability is the highest

function NumberofClusters(k_{max})
  $\bar{S}() := \{\}$ % List of normalized stability results per number of clusters
  for $k := 2$; $k <= k_{max}$ **do**
    S_k:= Stability(L, TC, P, I_s,r_{min}, Δr, ϵ_S) % Calculate stability
    S_k^R:= Stability(L, *Random*, P, I_s,r_{min}, Δr, ϵ_S) % Calculate random stability
    $\bar{S}_k := \frac{S_k - S_k^R}{1 - S_k^R}$ % Normalize with regards to random stability
  end for
  return $k := \underset{k}{\text{argmax}}(\bar{S}(k))$
end function

function Stability(L, TC, P, I_s,r_{min}, Δr, ϵ_S)
  $r := 1$ % Iteration
  $CL := TC(L)$ % Baseline clustered event log
  $u() := \{\}$ % List of similarity results per iteration
  $w() := \{\}$ % List of stability results per iteration
  while $(r < r_{min}) \vee [max_{p,q}|w(p) - w(q)| > \epsilon_S; \forall p, q : r - \Delta r < p < q \leq r)]$ **do**
    $L_r := P_r(L)$ % Perturb the log
    $CL_r := TC(L_r)$ % Clustered event log from perturbed log
    $u(r) := I_s(CL, CL_r)$ % Calculate similarity with baseline clustered event log
    $w(r) := \frac{(r-1)*w(r-1)+u(r)}{r}$ % Calculate stability
    $r := r + 1$
  end while
  return $S := w(r-1)$
end function

3.5 Step 5: Normalization of the stability

The final step is the normalization of the stability. This normalization is included to exclude unwanted information from entering the stability metric: if the random stability increases for higher cluster numbers, for example, than this is due to the inherent structure of the stability metric, rather than an actual improvement in the quality of the clustering. As provided in Algorithm 1, this is done as follows, where S_k is the stability of a certain clustering algorithm with k clusters, and S_k^R is the stability of randomly dividing the event log into clusters:

$$\bar{S}_k = \frac{S_k - S_k^R}{1 - S_k^R} \tag{6}$$

Finally, remark that a random clustering should cluster event logs based on their distinct process instances, not process instances. The underlying assumption is that any existing trace clustering technique should at least group those traces together that contain exactly the same behaviour, even a random clustering technique.

4 Experimental evaluation

This evaluation serves multiple purposes: first, it is meant to show the general applicability of our technique. Therefore, our approach is tested on multiple real-life datasets in combination with a wide variety of trace clustering techniques. Furthermore, the purpose is to evaluate the different components of our stability framework: the underlying resampling strategies, the similarity metrics, and the normalization.

4.1 Setup

This section describes the different event logs and trace clustering techniques that are used, and the components of our approach: how the perturbation will be applied; which similarity indices will be used for measuring the similarity between the baseline clustering and the clusterings on the perturbed event logs.

Four real-life event logs [5] are subjected to our approach. The number of process instances, distinct process instances, number of distinct events and average number of events per process instance are listed in Table 1. Observe that no exact number of clusters is known upfront for these event logs: the starting point is that applying process mining methods such as process discovery techniques on the entire event log leads to undesirable results [5]. Hence, this evaluation shows how our stability measure can be used to determine an appropriate number of clusters, or how it can be used to show that no appropriate number of clusters can be found.

With regards to trace clustering techniques, we have calculated the results using 7 different methods: 2 methods based on 'process-model aware' clustering techniques (*ActFreq* and *ActMRA*, [6]), and 5 'trace featurization' methods (*MR* and *MRA* [17]; *GED* an *LED* [2]; and *K-gram* [23]).[2]

With regards to the calculation of the stability, we have chosen to apply two strategies. On the one hand, a noise-induction perturbation strategy, where each process instance has a 10% chance of either having an event removed, two events swapped, or one event added from the existing activity alphabet. On the other hand, a sub-sampling approach, where 25% of the distinct process instances is removed. The probability of removal a distinct process instance is inversely proportional with its frequency in the event log.

Furthermore, both the Normalized Mutual Information similarity-metric (I_{NMI}) and the symmetrical discovered process model similarity metric based on Heuristics miner and graph-edit distance ($\bar{I}_{HG}$) will be employed, as described in Section 3.3. This allows for a comparison of the results of both similarity metrics.

Finally, the maximum number of clusters is set to 10. In addition, the evaluation strategy proposed in Algorithm 1 will deliberately not be used, to prevent randomization bias. Rather, a fixed number of 20 iterations will be used to calculate the stability, with appropriate seeding to prevent bias.

[2] The first two methods are implemented in the ProM-framework for process mining in the *ActiTrac*-plugin. The latter five methods are implemented in the *GuideTreeMiner*-plugin.

Table 1. Characteristics of the real-life event logs used for the evaluation: number of process instances (**#PI**), distinct process instances (**#DPI**), number of different events (**#EV**) and average number of events per process instance ($\frac{\#EV}{PI}$).

Log name	**#PI**	**#DPI**	**#EV**	$\frac{\#EV}{PI}$
MOA	2004	71	49	6.20
ICP	6407	155	18	5.99
MCRM	956	212	22	11.73
KIM	1541	251	18	5.62

4.2 Results

The results are presented in Table 2, which contains the number of clusters with maximal stability for each combination of similarity metric and perturbation strategy; in Figure 3, which visualises the results on the KIM-dataset; and Figure 4, which visualises the results on the ICP-dataset. Since no clear cluster structures were found for the MCRM- and MOA-datasets, these Figures are not included here[3]. Note that this does not imply a shortcoming of our approach, these event logs most likely simply do not contain relevant trace clusters.

Table 2. Number of clusters for which the normalized stability is maximal. Two different similarity metrics, two different perturbation strategies and seven different clustering techniques were used, on four real-life datasets. The number of clusters for which the stability would have been maximal if no normalization had been applied is included between brackets.

		Noise Induction				Sub-Sampling			
Similarity	Technique	KIM	MCRM	MOA	ICP	KIM	MCRM	MOA	ICP
I_{NMI}	ActFreq	4(5)	2(2)	2(10)	2(5)	4(4)	2(2)	4(10)	3(3)
I_{NMI}	ActMRA	2(4)	3(3)	7(7)	4(9)	4(9)	6(6)	3(10)	4(4)
I_{NMI}	GED	7(7)	3(3)	7(7)	3(3)	9(10)	2(4)	5(7)	3(3)
I_{NMI}	LED	4(10)	4(4)	2(2)	2(10)	7(8)	4(4)	3(9)	9(10)
I_{NMI}	MR	2(10)	2(2)	5(5)	10(10)	2(10)	2(2)	3(3)	10(10)
I_{NMI}	MRA	2(10)	2(2)	2(2)	4(5)	2(10)	2(2)	2(5)	4(5)
I_{NMI}	K-gram	2(10)	10(10)	2(9)	2(10)	2(10)	10(10)	2(10)	10(10)
$\bar{I}_{HG}$	ActFreq	3(7)	2(2)	10(2)	2(2)	4(4)	4(3)	9(10)	5(3)
$\bar{I}_{HG}$	ActMRA	3(3)	3(2)	7(2)	2(2)	4(4)	6(6)	10(10)	9(9)
$\bar{I}_{HG}$	GED	3(2)	2(2)	10(2)	6(2)	2(2)	10(2)	5(10)	10(10)
$\bar{I}_{HG}$	LED	3(2)	2(2)	3(2)	8(2)	4(2)	2(2)	9(10)	5(4)
$\bar{I}_{HG}$	MR	2(2)	2(2)	10(10)	6(2)	2(2)	2(2)	2(2)	10(10)
$\bar{I}_{HG}$	MRA	3(2)	2(2)	10(2)	5(5)	3(2)	4(2)	5(5)	2(2)
$\bar{I}_{HG}$	K-gram	3(2)	6(2)	6(2)	2(2)	4(2)	2(2)	2(2)	2(2)

[3] The visual representations of the MCRM- and MOA-event logs are available on *http://www.processmining.be/clusterstability/ATAEDResults*.

Similarity metrics. In Figures 3 an 4, the I_{NMI}-metric is presented on top row, while the $\bar{I}_{HG}$-metric is presented on the bottom row. For the KIM-dataset (Figure 3), no clear peaks are apparent in the plots with the results of the I_{NMI}-metric. In the results of the $\bar{I}_{HG}$-metric, a peak appears to be present at a cluster number of 3 when applying a noise induction-perturbation. Similarly, there appears to be a consensus about 3 or 4 clusters when applying a sub-sampling perturbation strategy. The same observation holds for dataset ICP (Figure 4): there appears to be a peak around 6 clusters when combining the $\bar{I}_{HG}$-metric with noise-induction, while no peaks are apparent for the I_{NMI}-metric. These findings are supported by Table 2.

Perturbation strategy. With regards to perturbation strategy, similar results are found on the KIM-dataset (Figure 3) regardless of whether noise induction or sub-sampling is applied. On the ICP-dataset (Figure 4), the results seem to be in favour of a noise-induction approach.

Normalization. Table 2 contains the number of clusters for which the normalized stability was maximal. The number of clusters for which the stability would have been maximal if no normalization had been applied is included between brackets. With regards to the stability without normalization, 16 of the 28 combinations combining noise induction with the $\bar{I}_{HG}$-metric would have had different best cluster numbers if no normalization had been applied. Likewise, 13 out of 28 results combining the I_{NMI}-metric with noise induction would have been different if no normalization had been applied. For sub-sampling, there would have been 11 and 15 differences with $\bar{I}_{HG}$ and I_{NMI}, respectively. The non-normalized application of $\bar{I}_{HG}$ generally leads to smaller cluster numbers, whereas the non-normalized application of I_{NMI} generally leads to higher cluster numbers. This validates the usefulness of the normalization: it prevents the results from favouring smaller (as with the $\bar{I}_{HG}$-metric) or larger numbers of clusters (as with the I_{NMI}-metric).

Finally, observe from Figures 3 and 4 that a lot of the normalized stability results are negative, especially when combining sub-sampling with I_{NMI} or noise induction with $\bar{I}_{HG}$. This means that these results are less stable than a random clustering. When combining noise induction with $\bar{I}_{HG}$ on the ICP-dataset, for example, the clusterings obtained using the *GED* or *LED* clustering techniques are lower than zero for each clustering number, implying that they behave less stable than a random clustering technique regardless of the number of clusters.

5 Discussion and future work

In this paper, an approach for determining an appropriate number of trace clusters is presented. All components of the approach are discussed in detail, and it is evaluated on four real-life datasets. This evaluation shows that utilizing a process model-based metric as underlying similarity metric leads to more desirable

results than using a consensus-based similarity metric. This implies that model-driven evaluation of trace clustering techniques is useful, supporting the claims of [6]. Furthermore, it is shown that log-perturbation based on noise induction slightly outperforms log-perturbation based on sub-sampling in this context. Finally, the importance of normalizing the stability with regards to the stability of a random clustering is illustrated.

With regards to future work, a couple of options exist. First, it could be useful to validate our approach in situations where expert knowledge about the number of trace clusters is present. For the four datasets we utilized, no such knowledge was available. In addition, expert knowledge could even be incorporated in a trace clustering approach. Secondly, certain clustering approaches, like *GED* and *K-Gram*, were shown to behave in a rather unstable manner, with lower stability than a random clustering. The cause of this instability should be investigated more thoroughly, as the perturbation used for resampling is most likely the cause of this instability: such techniques are likely quite sensitive to noise or incompleteness, and thus inherently less suited to real-life applications. To remedy this, techniques from the consensus clustering domain could be useful to create clustering ensembles, which are expected to behave in a more stable manner. Finally, the underlying similarity metric that was shown to behave the most desirably was the process model-based $\bar{I}_{HG}$-metric. However, it is built upon a process similarity metric, the graph-edit distance, that was not created with the purpose of comparing discovered process models in mind. In future work, a process similarity metric could be conceived that is tailored specifically towards this objective.

References

1. Bose, R.P.J.C., Van Der Aalst, W.M.P.: Trace clustering based on conserved patterns: Towards achieving better process models. In: Lect. Notes Bus. Inf. Process. vol. 43 LNBIP, pp. 170–181 (2010)
2. Bose, R., Aalst, W.V.D.: Context Aware Trace Clustering: Towards Improving Process Mining Results. Sdm pp. 401–412 (2009)
3. Davison, A.C., Hinkley, D.V.: Bootstrap methods and their application, vol. 1. Cambridge university press (1997)
4. De Medeiros, A.K.A., Weijters, A.J.M.M., Van Der Aalst, W.M.P.: Genetic process mining: An experimental evaluation. Data Min. Knowl. Discov. 14(2), 245–304 (2007)
5. De Weerdt, J., De Backer, M., Vanthienen, J., Baesens, B.: A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs. Inf. Syst. 37(7), 654–676 (2012)
6. De Weerdt, J., Vanden Broucke, S., Vanthienen, J., Baesens, B.: Active trace clustering for improved process discovery. IEEE Trans. Knowl. Data Eng. 25(12), 2708–2720 (2013)
7. Delias, P., Doumpos, M., Grigoroudis, E., Manolitzas, P., Matsatsinis, N.: Supporting healthcare management decisions via robust clustering of event logs. Knowledge-Based Syst. 84, 203–213 (2015)

8. Di Ciccio, C., Mecella, M., Mendling, J.: The effect of noise on mined declarative constraints. In: Ceravolo, P., Accorsi, R., Cudre-Mauroux, P. (eds.) Data-Driven Process Discovery and Analysis: Third IFIP WG 2.6, 2.12 International Symposium, SIMPDA 2013, Riva del Garda, Italy, August 30, 2013, Revised Selected Papers. pp. 1–24. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
9. Dijkman, R., Dumas, M., Van Dongen, B., Krik, R., Mendling, J.: Similarity of business process models: Metrics and evaluation. Inf. Syst. 36(2), 498–516 (2011)
10. van Dongen, B.F., Dijkman, R.M., Mendling, J.: Measuring Similarity between Business Process Models. Adv. Inf. Syst. Eng. 5074, 450–464 (2008)
11. Ekanayake, C.C., Dumas, M., García-Bañuelos, L., La Rosa, M.: Slice, mine and dice: Complexity-aware automated discovery of business process models. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics) 8094 LNCS, 49–64 (2013)
12. Ferreira, D., Zacarias, M., Malheiros, M., Ferreira, P.: Approaching Process Mining with Sequence Clustering: Experiments and Findings. LNCS 4714, 360–374 (2007)
13. Folino, F., Greco, G., Guzzo, A., Pontieri, L.: Editorial: Mining Usage Scenarios in Business Processes: Outlier-aware Discovery and Run-time Prediction. Data Knowl. Eng. (2011)
14. Fred, A., Lourenço, A.: Cluster ensemble methods: From single clusterings to combined solutions. Stud. Comput. Intell. 126, 3–30 (2008)
15. Goedertier, S., Martens, D., Vanthienen, J., Baesens, B.: Robust Process Discovery with Artificial Negative Events. J. Mach. Learn. Res. 10, 1305—-1340 (2009)
16. Greco, G., Guzzo, A., Pontieri, L., Saccà, D.: Discovering expressive process models by clustering log traces. IEEE Trans. Knowl. Data Eng. 18(8), 1010–1027 (2006)
17. Jagadeesh Chandra Bose, R.P., Van Der Aalst, W.M.P.: Abstractions in process mining: A taxonomy of patterns. In: Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). vol. 5701 LNCS, pp. 159–175 (2009)
18. Lange, T., Roth, V., Braun, M.L., Buhmann, J.M.: Stability-based validation of clustering solutions. Neural Comput. 16(6), 1299–1323 (2004)
19. Lee, Y., Lee, J.H., Jun, C.H.: Validation measures of bicluster solutions. Ind. Eng. Manag. Syst. 8(2), 101–108 (2009)
20. Lee, Y., Lee, J., Jun, C.H.: Stability-based validation of bicluster solutions. Pattern Recognit. 44(2), 252–264 (2011)
21. Maruster, L.: A machine learning approach to understand business processes. Eindhoven University of Technology (2003)
22. Mirkin, B.: Choosing the number of clusters. Wiley Interdiscip. Rev. Data Min. Knowl. Discov. 1(June), 252–260 (2011)
23. Song, M., Günther, C., van der Aalst, W.M.: Trace Clustering in Business Process Mining. In: Bus. Process Manag. Work. vol. 17, pp. 109–120. Springer (2009)
24. Tibshirani, R., Walther, G., Hastie, T.: Estimating the number of clusters in a data set via the gap statistic. J. R. Stat. Soc. Ser. B (Statistical Methodol. 63, 411–423 (2001)
25. Van der Aalst, W., Adriansyah, A., Van Dongen, B.: Replaying history on process models for conformance checking and performance analysis. Wiley Interdiscip. Rev. Data Min. Knowl. Discov. 2(2), 182–192 (2012)
26. Weidlich, M., Polyvyanyy, A., Desai, N., Mendling, J., Weske, M.: Process compliance analysis based on behavioural profiles. Inf. Syst. 36(7), 1009–1025 (2011)
27. a.J.M.M. Weijters, van der Aalst, W.: Rediscovering Workflow Models from Event-Based Data using Little Thumb. Integr. Comput. Eng. 10, 151–162 (2003)

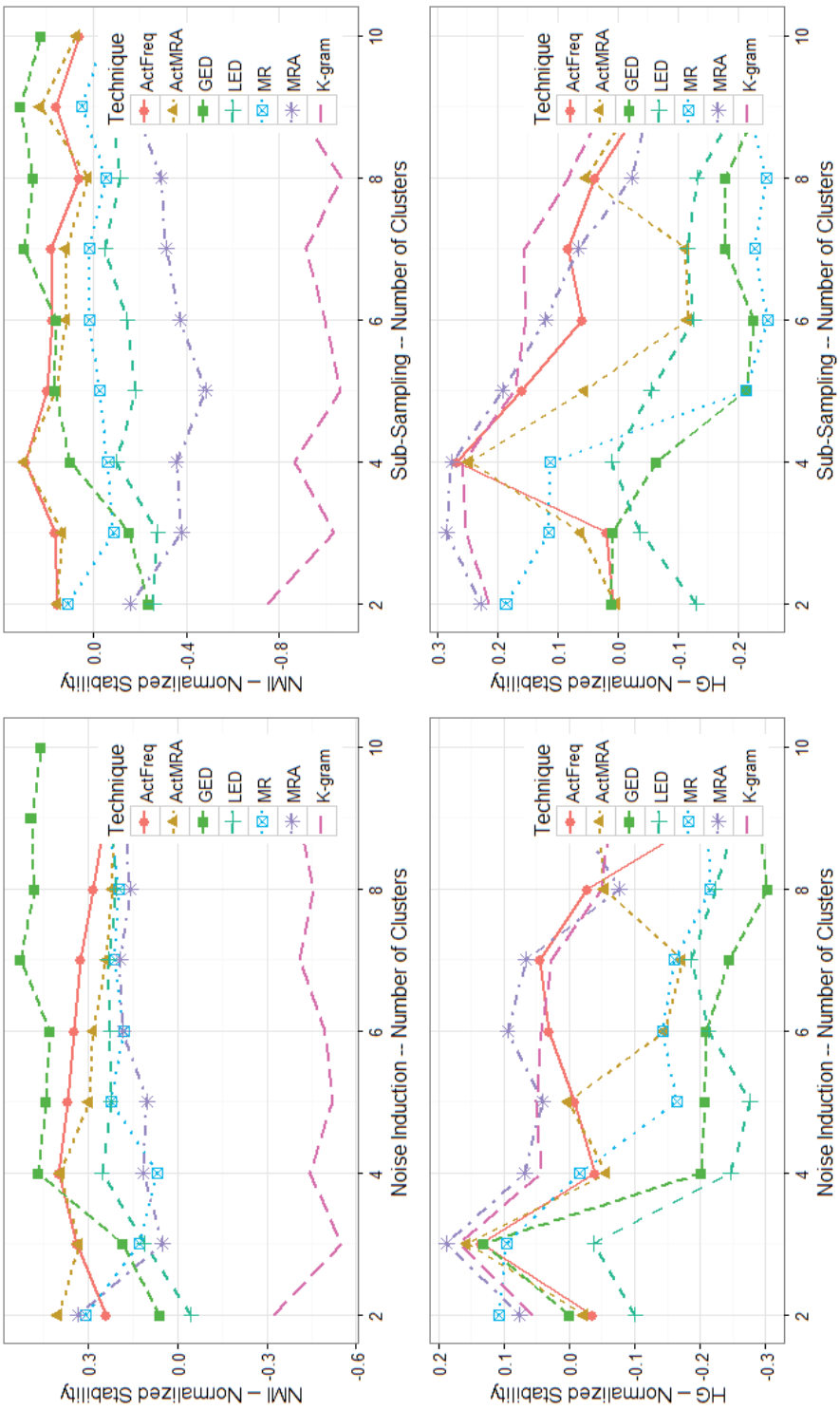

Fig. 3. Plot of the normalized stability results on the KIM-dataset in terms of the number of clusters, calculated with similarity metric I_{NMI} on the top row and $\bar{I}_{HG}$ on the bottom row. The results on the left are calculated with noise induction, the results on the right with sub-sampling.

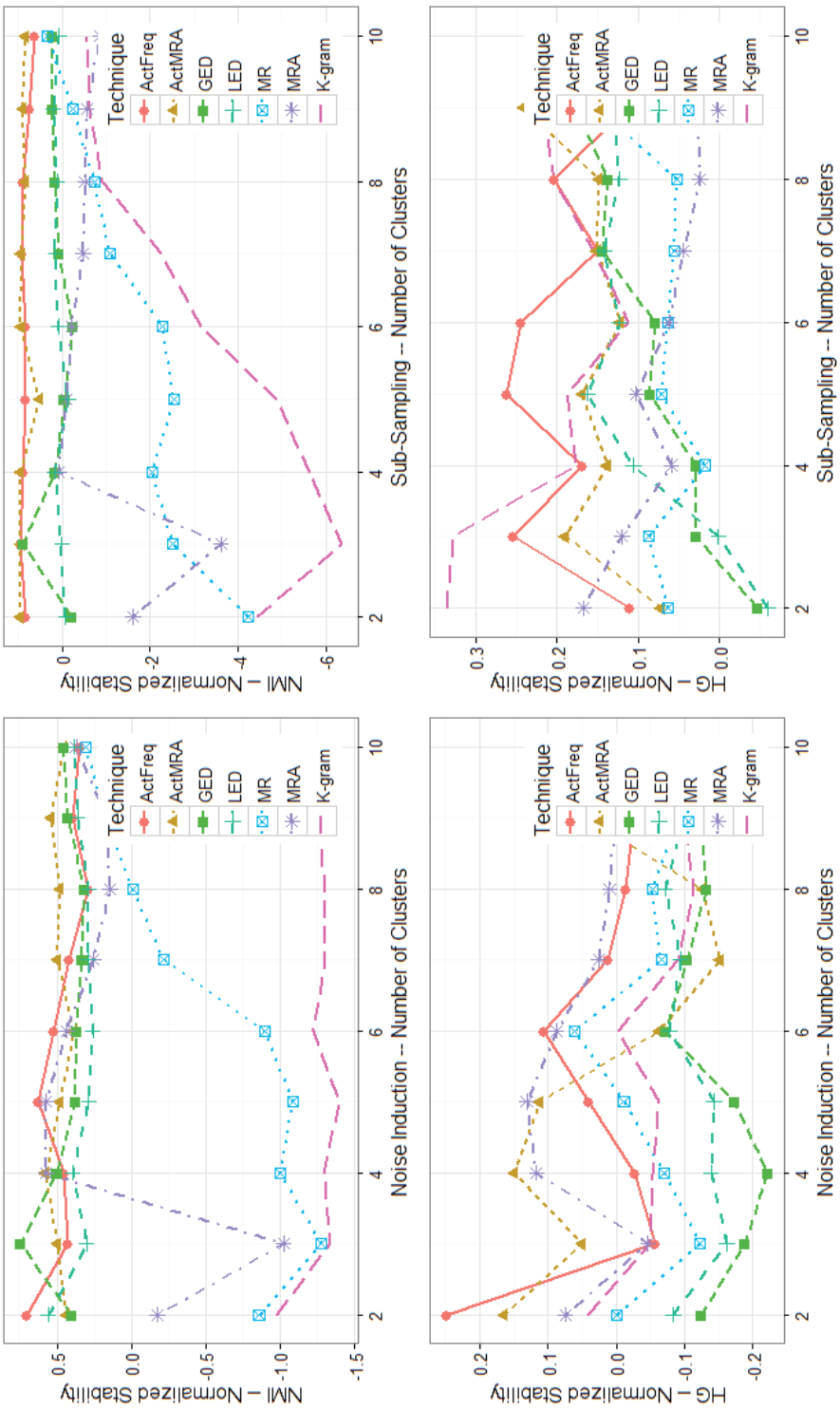

Fig. 4. Plot of the normalized stability results on the ICP-dataset in terms of the number of clusters, calculated with similarity metric I_{NMI} on the top row and $\bar{I}_{HG}$ on the bottom row. The results on the left are calculated with noise induction, the results on the right with sub-sampling.