

Similarity-Based Approaches for Determining the Number of Trace Clusters in Process Discovery

Pieter De Koninck^(✉) and Jochen De Weerd^{}

Research Center for Management Informatics,
Faculty of Economics and Business, KU Leuven, Leuven, Belgium
`pieter.dekoninck@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 presents approaches to determine the appropriate number of clusters in a trace clustering context. In order to fulfil the objective of identifying the most appropriate number of trace clusters, two approaches built on similarity are proposed: a stability- and a separation-based method. The stability-based method iteratively calculates the similarity between clustered versions of perturbed and unperturbed event logs. Alternatively, an approach based on between-cluster dissimilarity, or separation, is proposed. Regarding practical validation, both approaches are tested on multiple real-life datasets to investigate the complementarity of the different components. Our results suggest that both methods are successful in identifying an appropriate number of trace clusters.

Keywords: Stability · Trace clustering · Validity · Log perturbation · Process discovery · Separation

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 [29], 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 approaches 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 first 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 this 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 clusters¹.

Our second approach is based on the concept of separation of a clustering solution. Conceptually, one prefers a clustering solution where the clusters are well separated, i.e. where the clusters are not too similar. For this, a component of the first stability-based approach, a method for calculating the similarity between trace clustering solutions, is leveraged to capture the separation of a clustering solution. Like the stability-based approach, it is evaluated on four real-life event logs.

The remainder of this article is structured as follows: in Sect. 2, the necessary background on the process mining domain is given, as well an overview of existing approaches for determining the number of clusters. Our stability-based approach is outlined in Sect. 3, while Sect. 4 details the separation-based approach. Finally, both approaches are evaluated in Sect. 5, before finishing with some concluding remarks in Sect. 6.

2 Background

This section contains the necessary background on the domain of process mining, as well as an overview of existing general approaches for determining the number of clusters in traditional clustering.

2.1 Event Logs, Process Discovery and Trace Clustering

Trace clustering, as it is considered in this paper, is a part of the process mining domain. Generally speaking, process mining consists of three distinct parts:

¹ This approach is implemented as an experimental ProM-plugin which can be found on <http://www.processmining.be/clusterstability/>.

process discovery, conformance checking, and process enhancement [1]. In process discovery, the starting point is an event log L , from which one wants to discover a corresponding process model M . Typically, this event log adheres to the IEEE eXtensible Event Stream (XES) standard². In conformance checking, an event log L containing actual behaviour, and a process model M containing prescribed behaviour are compared to detect deviations between expected and observed behaviour. Finally, process enhancement is an umbrella term for techniques that aim to improve processes, for example by suggesting improvements to the as-is process-model.

One of the main problems surrounding the application of process discovery techniques to real-life datasets, however, is that they typically contain a wide variety of behaviour. Applying conventional process discovery techniques on event logs that contain such variation will most likely lead to sub-optimal results [8]. Therefore, a variety of authors [2, 9, 16] have proposed to apply clustering on event logs in order to improve the quality of the process models that can be mined from these event logs. Since an event log is a set of traces, this sub-discipline is called trace clustering.

2.2 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 [26]. 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 [26], 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 [28], 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 [26]. 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

² For more info on the XES-standard, we refer to <http://www.xes-standard.org/>.

is based on resampling, and is related to consensus clustering in its intuition: a number of 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 [9], this is not deemed an appropriate route for trace clustering. Rather than using direct trace similarity, a structure-based approach building on the concept of separation is presented in this paper. Consensus- and resampling-based approaches do not suffer from this issue as much. Therefore, a consensus- and resampling-based approach leveraging the concept of stability is described in this paper as well.

3 Stability of Trace Clustering Solutions

The approach proposed in this section is a resampling-based approach, inspired by a methodology for stability-based validation of clustering solutions in [22], which was adapted for biclustering solutions in [24]. In [22], it was shown to be an effective method for discovering the appropriate number of clusters on simulated and gene expression data. Furthermore, in [6], a similar stability-based approach is proposed to assess the quality of process discovery techniques.

In [22, 24], 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 than 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 Fig. 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 to P_n .
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: $CL_i = TC(P_i)$ 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 $I(CL, CL_i)$ for each $i \in \{1..n\}$.

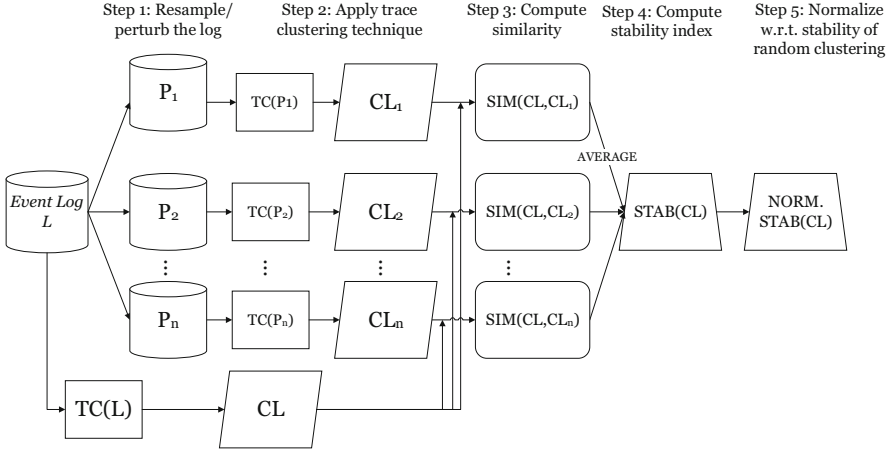


Fig. 1. A visualization of the proposed approach for calculating the stability of a clustered event log, based on a similar diagram in [24]. The normalized version of this stability is calculated at different numbers of clusters to determine the optimal number of clusters.

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

$$S^{TC} = \frac{1}{n} \sum_{i=1}^n I(CL, CL_i) \quad (1)$$

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

$$\bar{S}^{TC} = \frac{S^{TC} - S^R}{1 - S^R} \quad (2)$$

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 (Sect. 3.1), computation of solution similarity based on mutual information or process model similarity metrics (Sect. 3.3), calculation of the stability index based on a window-based approach (Sect. 3.4), and normalization of the stability with respect to random stability (Sect. 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 [26]: 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. Note that here, cases would be process instances or traces. When dealing with event logs, an important consideration is whether to sample based on process instances or distinct process instances. 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 [25], four types of noise were initially defined: remove head, remove tail, remove body, and swap tasks. In [7], 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 [11, 19].

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 of this distinct process instance in the event log. Secondly, behaviour can be added through noise induction. Though several noise types were proposed in [25], 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. Example 1 presents the effects of noise induction on a single 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 needs to be chosen.

Example 1. Given a process instance $ABBCD$, some potential effects of noise induction could be removal: $ABCD$, swapping: $ABCBD$, or addition: $ABBCCD$.

3.2 Step 2: Trace Clustering Technique

In the next step, a certain trace clustering algorithm is applied. An overview of existing trace clustering techniques is provided in Table 1. In general, trace clustering techniques can be classified according to three dimensions. Firstly, what they consider as input: a propositional representation or an event log. Secondly, which kind of clustering approach is used: for example k-means, hierarchical, model-driven. Thirdly and most importantly, the clustering bias they employ. Two broad categories exist: those that map traces onto a vector space model or quantify the similarity between two traces directly; and those that take the quality of the underlying process models into account.

Table 1. Existing trace clustering approaches with their data representation, clustering approaches and biases

Author	Data representation	Clustering technique	Clustering bias
Greco et al. [20]	Propositional	k-means	Instance similarity: alphabet k-grams
Song et al. [27]	Propositional	Various	Instance similarity: <i>profiles</i>
Ferreira et al. [16]	Event log	First order Markov mixture model	Maximum likelihood
Bose and van der Aalst [2,3]	Event log	Hierarchical clustering	Instance similarity metrics
Folino et al. [17]	Event log	Enhanced Markov cluster model	Maximum likelihood
De Weerd et al. [9]	Event log	Model-driven clustering	Combined process model fitness (ICS)
Ekanayake et al. [14]	Propositional	Complexity-aware clustering	Instance similarity + repository complexity
Delias et al. [10]	Event log	Spectral	Robust instance similarity
Evermann et al. [15]	Event log	k-means	Instance similarity: alignment cost

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 [18]. 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 to which two process instances are clustered together 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:

$$I_{NMI}(a, b) = -2 \frac{\sum_{i=1}^{k_a} \sum_{j=1}^{k_b} n_{ij}^{ab} \log\left(\frac{n_{ij}^{ab} n}{n_i^a n_j^b}\right)}{\sum_{i=1}^{k_a} n_i^a \log\left(\frac{n_i^a}{n}\right) + \sum_{j=1}^{k_b} n_j^b \log\left(\frac{n_j^b}{n}\right)} \quad (3)$$

On the other hand, we propose a metric based on the similarity between discovered process models. Rather than measuring 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 Fig. 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 [31]. It mines a heuristic net, which is converted to a Petri net. This technique has demonstrated its efficacy on real-life datasets [8]. With regards to process model similarity, our preference goes out to the structural graph-edit distance (GED) similarity metric [12], though behavioural metrics such as causal footprints [13] or behavioural profiles [30] could be used as well. Graph-edit distance is a metric that reflects the distance between two models based on the insertion and deletion of places and transitions in a Petri net. Finally, our similarity metric for trace clustering solutions is summarized in Eq. 4, where 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 .

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

In [23], 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 should be symmetric. Note from Eq. 3 that these properties are fulfilled for $I_{NMI}(\cdot, \cdot)$. Likewise, from Eq. 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} :

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

Example 2. Figure 3 contains two clustering solutions, each consisting of 2 clusters. To calculate $I_{HG}(a, b)$ for each cluster in clustering A, the most similar cluster in cluster B is to be chosen, and then the corresponding similarities are weighted using Eq. 4. Both clusters are most similar to cluster 1 of clustering B, resulting in a $I_{HG}(a, b)$ of 0.9. Assuming that both clusters in clustering B contain an equal amount of traces, $I_{HG}(b, a)$ is equal to 0.83, and $\bar{I}_{HG}(a, b)$ is 0.87.

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.

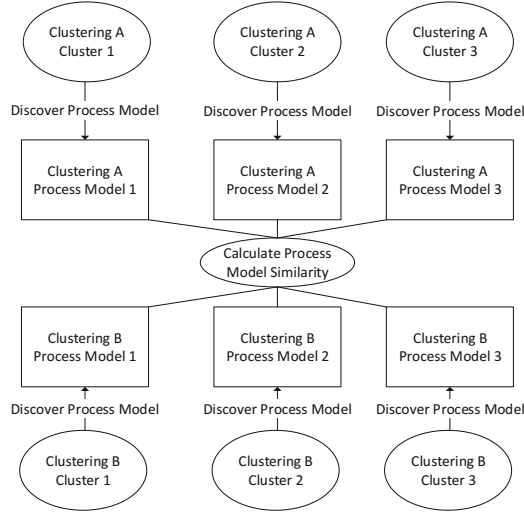


Fig. 2. Conceptual representation of the process model-based similarity metric, when two clusterings of three clusters each are under comparison.

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. The approach goes as follows (lines 2–8): for each of the number of clusters, the stability is calculated, as is the stability of a random clustering technique, which is then used to calculate the normalized stability. The number of clusters with the highest stability is returned. The stability (lines 10–22) is calculated by clustering the entire log to create a baseline clustered log. Then, the log is iteratively perturbed, clustered, and the similarity between this clustered log and the baseline clustered log is computed. The stability is calculated as the average of these similarities over the iterations. When the minimum amount of iterations (r_{min}) has passed, and the stability has not deviated more than the maximal error (ϵ_S) in the review window (last Δr iterations), the stability-function terminates.

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, then 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

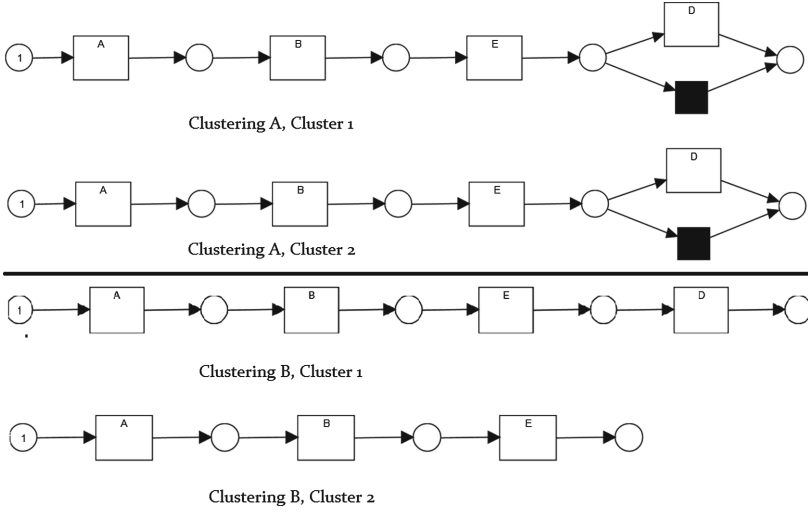


Fig. 3. Example of two clustering solutions and the process models corresponding to each cluster.

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:

$$\bar{S}_k = \frac{S_k - S_k^R}{1 - S_k^R} \quad (6)$$

Example 3. Given a hypothetical situation where one is determining whether 3 or 5 clusters is most appropriate, the normalization could have the effect illustrated in Table 2, reducing the preference for a number of clusters where even a random clustering is stable.

Table 2. Example of effect of the normalization.

Clusters	S_k	S_k^R	\bar{S}_k
3	0.6	0.5	0.2
5	0.6	0.4	0.33

Finally, we 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.

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

```

1: function NUMBEROFCLUSTERS(  $k_{max}$  )
2:    $\bar{S}() := \{\}$  % List of normalized stability results per number of clusters
3:   for  $k := 2$  ;  $k \leq k_{max}$  do
4:      $S_k :=$  STABILITY(  $L, TC, P, I_s, r_{min}, \Delta r, \epsilon_S$  ) % Calculate stability
5:      $S_k^R :=$  STABILITY(  $L, Random, P, I_s, r_{min}, \Delta r, \epsilon_S$  ) % Calculate random stability
6:      $\bar{S}_k := \frac{S_k - S_k^R}{1 - S_k^R}$  % Normalize with regards to random stability
7:   end for
8:   return  $k := \operatorname{argmax}_k(\bar{S}(k))$ 

9: end function

10: function STABILITY(  $L, TC, P, I_s, r_{min}, \Delta r, \epsilon_S$  )
11:    $r := 1$  % Iteration
12:    $CL := TC(L)$  % Baseline clustered event log
13:    $u() := \{\}$  % List of similarity results per iteration
14:    $w() := \{\}$  % List of stability results per iteration
15:   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
16:      $L_r := P_r(L)$  % Perturb the log
17:      $CL_r := TC(L_r)$  % Cluster event log from perturbed log
18:      $u(r) := I_s(CL, CL_r)$  % Calculate similarity with baseline clustered event log
19:      $w(r) := \frac{(r-1) * w(r-1) + u(r)}{r}$  % Calculate stability
20:      $r := r + 1$ 
21:   end while
22:   return  $S := w(r - 1)$ 
23: end function

```

4 A Cross-Cluster Separation-Based Approach

As detailed in Sect. 2.2, a wide array of different possible directions exist when it comes to assessing the number of clusters. In the previous section, an approach based on stability has been detailed. In this section, one of the constructs used to calculate this stability, $I_{HG}(a, b)$, is leveraged to create an alternative path for assessing an appropriate number of trace clusters.

Conceptually, cross-cluster separation represents how well the clusters in a partitioning are separated. Clearly, a quantification of separation can be used as a metric for the quality of a cluster solution. The most well-known separation metric is probably the Davies-Bouldin metric [4]. It can be considered a structural approach in the taxonomy of [26]. The main issue when it comes to defining such a metric in the trace clustering case, however, lies in the definition of the similarity between two traces. This can be done by incorporating a direct similarity between the traces, based on alignment or based on a mapping of the traces, however, clustering techniques exist that specifically aim to avoid such a mapping. Therefore, in this section, we propose a similarity metric based on the process models that represent each cluster, as in Sect. 3.3. 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 similarity can then be used to calculate the separation of the clusters. While a number of process discovery techniques and process model similarity techniques

exist which could be used for this task, we apply the Heuristics miner and the structural graph-edit distance, as before.

More specifically, we propose to use an internal version of the $I_{HG}(a, b)$ similarity index as a measure of cluster separation, where 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 nets mined from clusters i and j of clustering a .

$$Sep_{HG}(a) = 1 - \frac{\sum_{i \in a} n_i \max_{(j \in a) \neq i} (sim_{HG}(i, j))}{\sum_{i \in a} n_i} \quad (7)$$

Three observations can be made regarding this weighted metric for inter-cluster separation based on the process model similarity of discovered process models. First, observe that it is inherently symmetrical, therefore no extra step is needed to render it symmetrical, in contrast to Eq. 5. Secondly, observe that this metric will be lower when inter-cluster separation is lower. Therefore, a higher value of the Sep_{HG} metric is preferable, and can be used as an indicator of an appropriate number of clusters.

Example 4. Figure 3 contains a comparison of 2 clustering solutions, both comprised of two clusters. In the top row, both clusters in the 2-cluster solution are represented by the same discovered process model. Therefore, the separation of this solution is 0. In the bottom row, the same set of traces is represented by two different discovered process models, with a separation of 0.16, assuming that both clusters contain an equal amount of process instances.

5 Experimental Evaluation

This evaluation serves multiple purposes: first, it is meant to show the general applicability of our techniques. Therefore, our approaches are 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. Finally, the separation-based approach is demonstrated here as well, and its results are contrasted with those of the stability-based approach.

5.1 Setup

This section describes the different event logs and trace clustering techniques that are used, and the components of the stability-based 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.

General. Four real-life event logs [8] 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 3. 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 [8]. Hence, this evaluation shows how our stability measure and separation measure can be used to determine an appropriate number of clusters, or how they 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*, [9]), and 5 ‘trace featurization’ methods (*MR* and *MRA* [21]; *GED* and *LED* [3]; and *K-gram* [27]).³

Stability. 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 Sect. 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.

Table 3. 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	$\frac{\#EV}{PI}$
KIM	1541	251	18	5.62
MCRM	956	212	22	11.73
MOA	2004	71	49	6.20
ICP	6407	155	18	5.99

Separation. With regards to separation, the non-normalized version of our proposed separation metric, Sep_{HG} , is used, as described in Sect. 4. This approach

³ 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 *GuideTree-Miner*-plugin.

is also tested on a number of clusters ranging from 2 to 10, on the same event logs and with the same clustering techniques as the stability-based approach.

5.2 Results of the Stability-Based Approach

The results are presented in Table 4, which contains the number of clusters with maximal stability for each combination of similarity metric and perturbation strategy; in Fig. 4, which visualises the results on the KIM-dataset; and Fig. 5, 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 here⁴. Note that this does not imply a shortcoming of our approach, these event logs most likely simply do not contain relevant trace clusters.

Table 4. 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.

Similarity	Technique	Noise induction				Sub-Sampling			
		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)

Similarity Metrics. In Figs. 4 and 5, the I_{NMI} -metric is presented in the top row, while the \bar{I}_{HG} -metric is presented in the bottom row. For the KIM-dataset (Fig. 4), no clear peaks are apparent in the plots with the results of the

⁴ The visual representations of the MCRM- and MOA-event logs are available on <http://www.processmining.be/clusterstability/ToPNoCResults>.

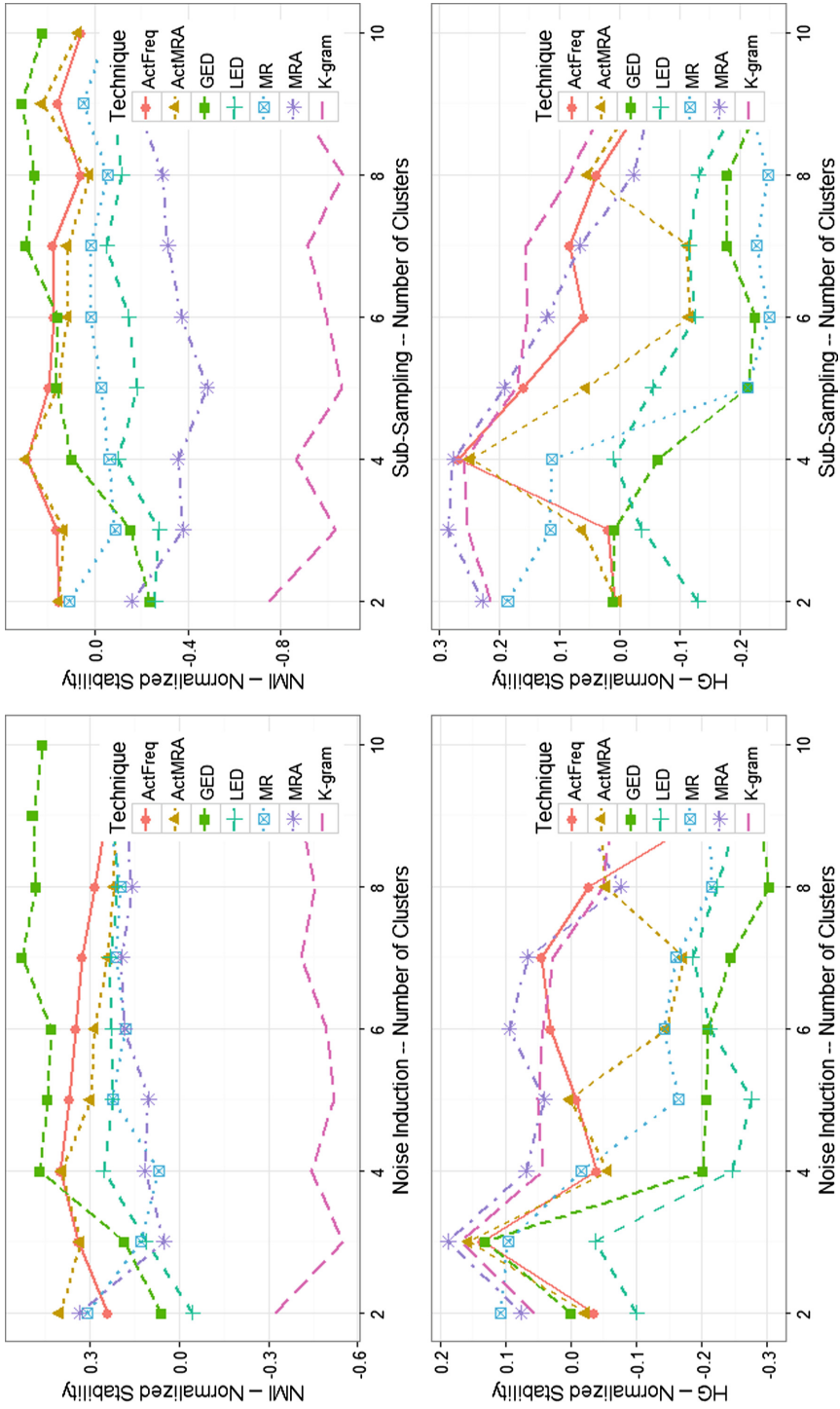


Fig. 4. Plot of the normalized stability results on the KIM-dataset in terms of the number of clusters, calculated with similarity metric I_{NMI} in the top row and I_{HG} in the bottom row. The results on the left are calculated with noise induction, the results on the right with sub-sampling.

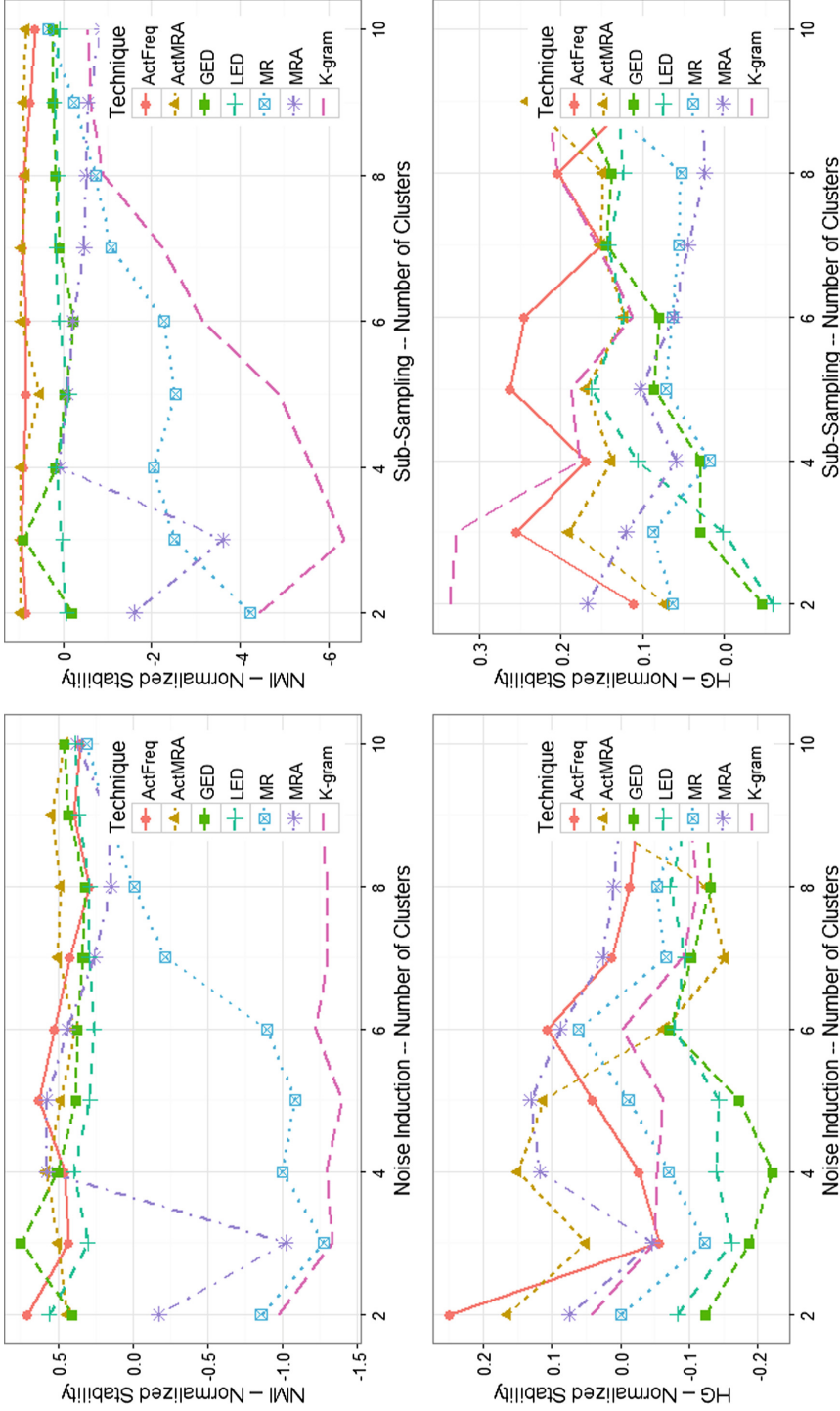


Fig. 5. Plot of the normalized stability results on the ICP-dataset in terms of the number of clusters, calculated with similarity metric I_{NMI} in the top row and \bar{I}_{HG} in the bottom row. The results on the left are calculated with noise induction, the results on the right with sub-sampling.

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 (Fig. 5): 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 4.

Perturbation Strategy. With regards to perturbation strategy, similar results are found on the KIM-dataset (Fig. 4) regardless of whether noise induction or sub-sampling is applied. On the ICP-dataset (Fig. 5), the results seem to be in favour of a noise-induction approach.

Algorithmic Efficiency. In Fig. 6, the evolution of the stability before normalization over the iterations is visualised, calculated with similarity metric \bar{I}_{HG} and noise induction, and using the K -gram clustering technique. This set-up is chosen as an example, with other configurations performing similarly. It is clear that the stability results converge rather quickly over the iterations, and that it was indeed appropriate to fix the number of iterations at 20 in the other evaluations in this section of the paper.

In Fig. 7, the evaluation of the computational time (in seconds) over the number of iterations is presented. These results were obtained in Java SE 8, on a device running Windows 10 Enterprise, with an Intel Core i7-4712HQ processor. The K -gram clustering technique is evaluated with similarity metric \bar{I}_{HG} and noise induction. As expected, the runtime increases linearly over the number of iterations, with the slope depending on the number of clusters under scrutiny (the more clusters, the higher). Recall that these are real-life dataset, and that the ICP-event log is rather large, see Table 3. Nonetheless, even at 40 iterations, the algorithm terminates in under 4 min.

Normalization. Table 4 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).

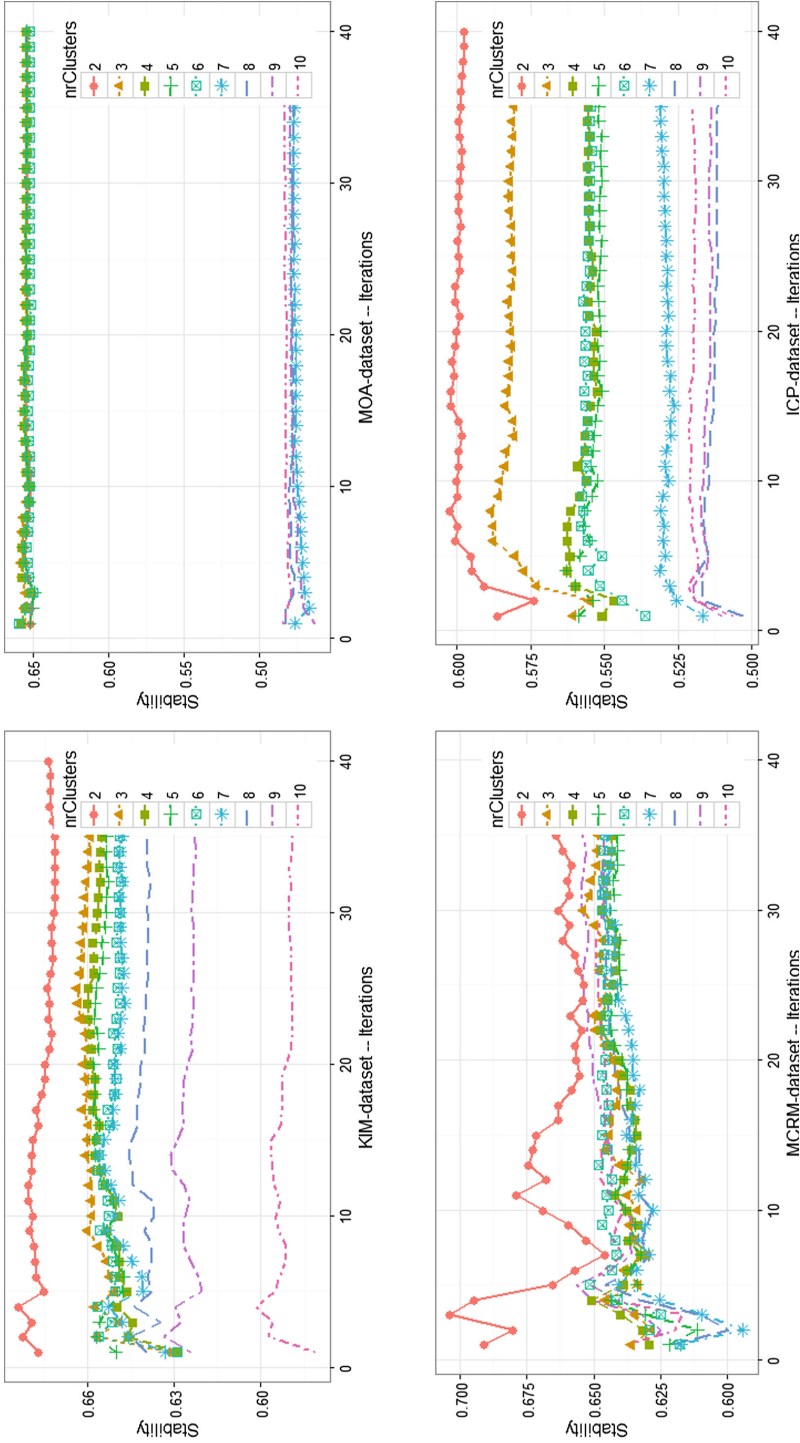


Fig. 6. Plot of the evolution of the stability results before normalization over the iterations on each dataset, calculated with similarity metric \bar{I}_{HG} and noise induction, using the K -gram clustering technique.

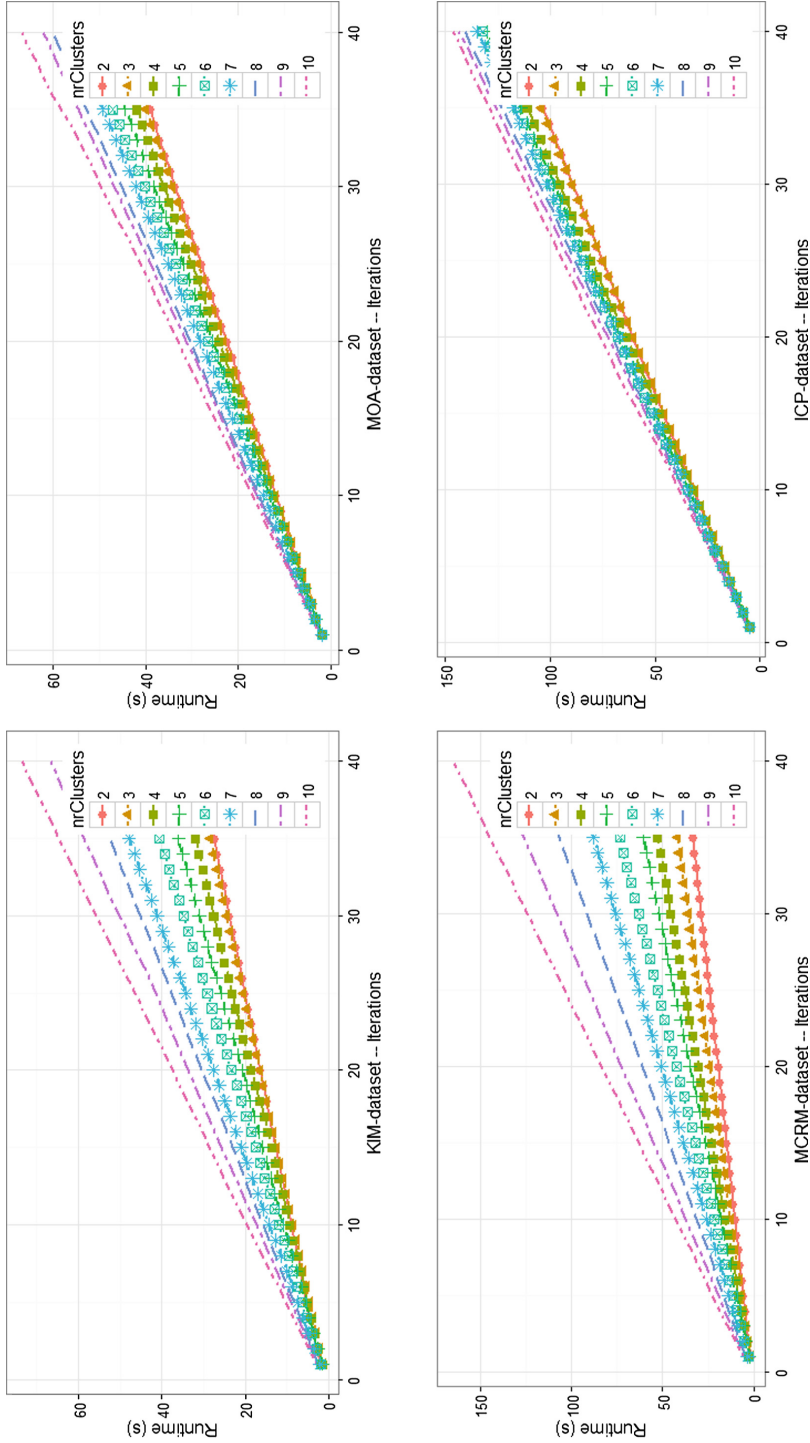


Fig. 7. Plot of the evolution of the runtime of the stability calculation over the iterations on each dataset, calculated with similarity metric \bar{I}_{HG} and noise induction, using the K -gram clustering technique.

Finally, observe from Figs. 4 and 5 that a lot of the normalized stability results are negative, especially when combining sub-sampling with I_{NMI} or noise induction with \tilde{I}_{HG} . This means that these results are less stable than a random clustering. When combining noise induction with \tilde{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.

5.3 Results of the Separation-Based Approach

The results of the separation-based approach are visualised in Fig. 8. For each dataset, the values of Sep_{HG} are plotted at a number of clusters ranging from 2 to 10, for 6 different trace clustering techniques. A couple of observations can be made from these plots. First, observe that most curves display a downward trend, indicating that the separation of the clusters in terms of process model similarity declines as the number of clusters increases. This is in line with the expectations one has when clustering a dataset in which no true clusters are present. Nonetheless, a different trend is present in the separation results of the ICP-event log. Specifically, the clustering solutions obtained when applying an *ActFreq*, *ActMRA* or *LED* clustering technique, score better on separation with 4 clusters than with 2 or 3 clusters. This is an indication that partitioning this event log into 4 clusters is appropriate for the ICP-data set.

Table 5. Number of clusters for which the separation metric Sep_{HG} is maximal. Seven different clustering techniques were used, on four real-life datasets.

Technique	KIM	MCRM	MOA	ICP
ActFreq	2	2	2	8
ActMRA	2	3	2	4
GED	2	3	2	2
LED	2	2	2	5
MR	2	3	2	2
MRA	2	2	2	2
K-gram	2	3	3	2

Finally, Table 5 contains the exact number of clusters at which the Sep_{HG} metric is maximal. It can be seen as a less nuanced version of Fig. 8. As such, there is again an indication of a true cluster presence in the application of the *ActFreq*, *ActMRA* or *LED* clustering techniques on the ICP-dataset, as well as an indication regarding the application of the *ActMRA*, *GED*, *MR* and *K-gram* on the MCRM-dataset, where three clusters appear to be optimal. On the KIM- and MOA-datasets, no indication of a cluster number higher than 2 is present, except for the application of the *K-gram* technique on the MOA-set.

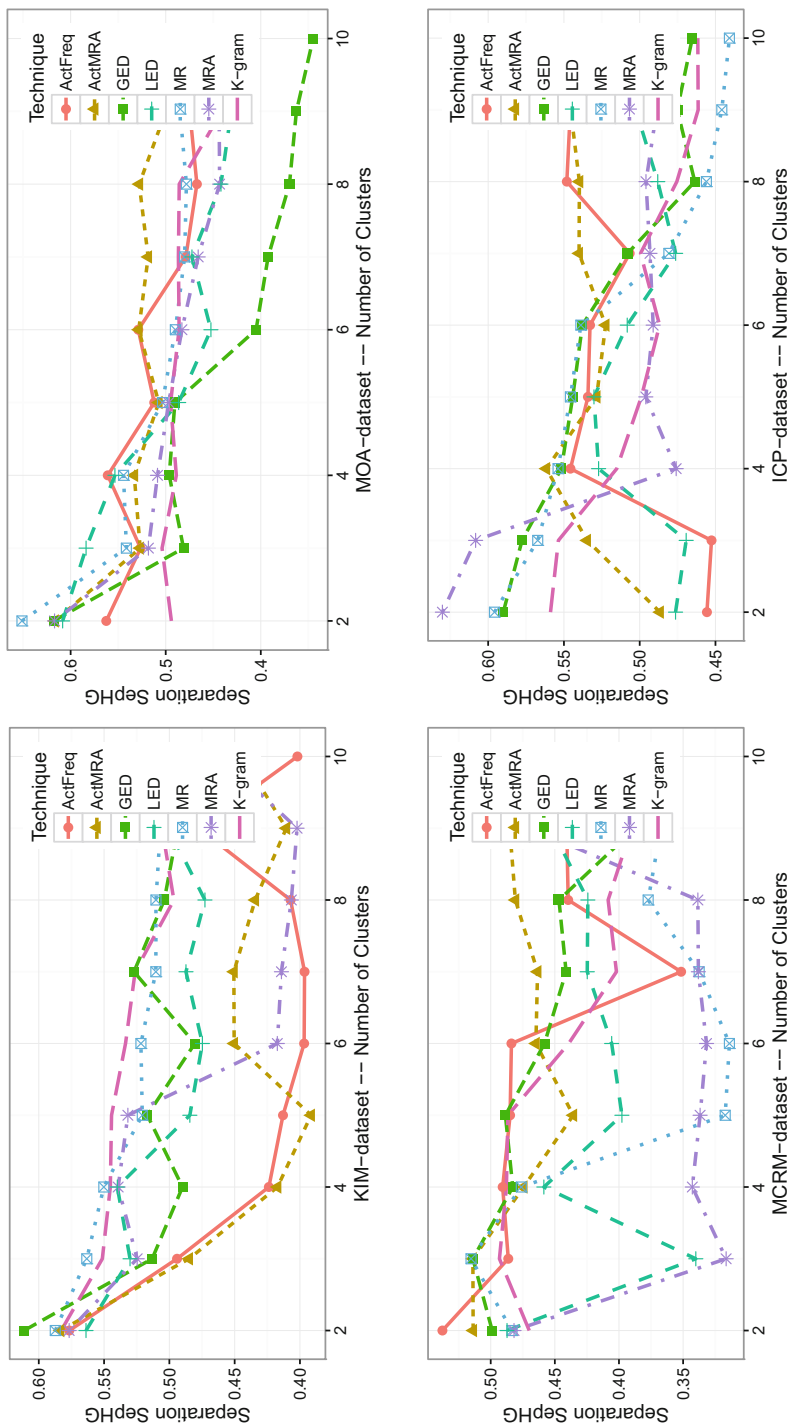


Fig. 8. The results of the separation-based approach on the four datasets.

6 Discussion and Future Work

In this paper, two approaches for determining an appropriate number of trace clusters is presented. The main contribution is a stability-based approach. All components of this 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 [9]. 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.

As a contrasting approach to the stability-based one, a separation-driven approach is proposed as well. It is based on process model-similarity, and is shown to be a useful alternative approach in the evaluation. Remarkably, the separation-based approach is shown to lead to different conclusions than the stability-based approach, suggesting that a true cluster structure is present in the MCRM-dataset and not in the KIM-dataset. Both approaches lead to similar conclusions on the two other datasets.

With regards to future work, some 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. Fourthly, the separation-based approach presented here could be used in a more traditional structure-based approach according to the taxonomy of [26]. To achieve this, the cross-cluster separation should be contrasted with the within-cluster cohesion of a trace clustering. Finally, combining the stability- and the separation-based approaches into a single, hybrid approach can be considered an interesting avenue for future work.

Note. This paper extends and enhances [5], in three distinct ways: (1) it builds on \bar{I}_{HG} to propose a separation-based approach for determining the number of traces clusters, (2) this approach is evaluated on real-life event logs, and (3) the number of iterations needed for the calculation of the stability and the needed computational time are evaluated. These extensions are contained mainly in Sects. 2.1, 4 and 5.

References

1. van der Aalst, W.: *Process Mining: Data Science in Action*. Springer, Berlin (2016)
2. Bose, R.P.J.C., van der Aalst, W.M.P.: Trace clustering based on conserved patterns: towards achieving better process models. In: Rinderle-Ma, S., Sadiq, S., Leymann, F. (eds.) *BPM 2009. LNBIP*, vol. 43, pp. 170–181. Springer, Heidelberg (2010). doi:[10.1007/978-3-642-12186-9_16](https://doi.org/10.1007/978-3-642-12186-9_16)
3. Bose, R., Aalst, W.V.D.: Context aware trace clustering: towards improving process mining results. In: *SDM*, pp. 401–412 (2009)
4. Davies, D.L., Bouldin, D.W.: A cluster separation measure. *IEEE Trans. Pattern Anal. Mach. Intell.* **2**, 224–227 (1979)
5. De Koninck, P., De Weerd, J.: Determining the number of trace clusters: a stability-based approach. In: *Proceedings of the International Workshop on Algorithms & Theories for the Analysis of Event Data (ATAED) 2016*, vol. 1592, pp. 1–15. *CEUR-ws Workshop Proceedings* (2016)
6. De Koninck, P., De Weerd, J.: A stability assessment framework for process discovery techniques. In: La Rosa, M., Loos, P., Pastor, O. (eds.) *BPM 2016. LNCS*, vol. 9850, pp. 57–72. Springer, Cham (2016). doi:[10.1007/978-3-319-45348-4_4](https://doi.org/10.1007/978-3-319-45348-4_4)
7. 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)
8. De Weerd, 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. *Inform. Syst.* **37**(7), 654–676 (2012)
9. De Weerd, 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)
10. 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)
11. Di Ciccio, C., Mecella, M., Mendling, J.: The effect of noise on mined declarative constraints. In: Ceravolo, P., Accorsi, R., Cudre-Mauroux, P. (eds.) *SIMPDA 2013. LNBIP*, vol. 203, pp. 1–24. Springer, Heidelberg (2015). doi:[10.1007/978-3-662-46436-6_1](https://doi.org/10.1007/978-3-662-46436-6_1)
12. Dijkman, R., Dumas, M., Van Dongen, B., Krik, R., Mendling, J.: Similarity of business process models: metrics and evaluation. *Inform. Syst.* **36**(2), 498–516 (2011)
13. van Dongen, B., Dijkman, R., Mendling, J.: Measuring similarity between business process models. In: Bellahsene, Z., Léonard, M. (eds.) *CAiSE 2008. LNCS*, vol. 5074, pp. 450–464. Springer, Heidelberg (2008). doi:[10.1007/978-3-540-69534-9_34](https://doi.org/10.1007/978-3-540-69534-9_34)
14. Ekanayake, C.C., Dumas, M., García-Bañuelos, L., La Rosa, M.: Slice, mine and dice: complexity-aware automated discovery of business process models. In: Daniel, F., Wang, J., Weber, B. (eds.) *BPM 2013. LNCS*, vol. 8094, pp. 49–64. Springer, Heidelberg (2013). doi:[10.1007/978-3-642-40176-3_6](https://doi.org/10.1007/978-3-642-40176-3_6)
15. Evermann, J., Thaler, T., Fettke, P.: Clustering traces using sequence alignment. In: Reichert, M., Reijers, H.A. (eds.) *BPM 2015. LNBIP*, vol. 256, pp. 179–190. Springer, Cham (2016). doi:[10.1007/978-3-319-42887-1_15](https://doi.org/10.1007/978-3-319-42887-1_15)
16. Ferreira, D., Zacarias, M., Malheiros, M., Ferreira, P.: Approaching process mining with sequence clustering: experiments and findings. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) *BPM 2007. LNCS*, vol. 4714, pp. 360–374. Springer, Heidelberg (2007). doi:[10.1007/978-3-540-75183-0_26](https://doi.org/10.1007/978-3-540-75183-0_26)

17. 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.* **70**, 1005–1029 (2011)
18. Fred, A., Lourenço, A.: Cluster ensemble methods: from single clusterings to combined solutions. *Stud. Comput. Intell.* **126**, 3–30 (2008)
19. Goedertier, S., Martens, D., Vanthienen, J., Baesens, B.: Robust process discovery with artificial negative events. *J. Mach. Learn. Res.* **10**, 1305–1340 (2009)
20. 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)
21. Jagadeesh Chandra Bose, R.P., van der Aalst, W.M.P.: Abstractions in process mining: a taxonomy of patterns. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) *BPM 2009. LNCS*, vol. 5701, pp. 159–175. Springer, Heidelberg (2009). doi:[10.1007/978-3-642-03848-8_12](https://doi.org/10.1007/978-3-642-03848-8_12)
22. Lange, T., Roth, V., Braun, M.L., Buhmann, J.M.: Stability-based validation of clustering solutions. *Neural Comput.* **16**(6), 1299–1323 (2004)
23. Lee, Y., Lee, J.H., Jun, C.H.: Validation measures of bicluster solutions. *Ind. Eng. Manag. Syst.* **8**(2), 101–108 (2009)
24. Lee, Y., Lee, J., Jun, C.H.: Stability-based validation of bicluster solutions. *Pattern Recognit.* **44**(2), 252–264 (2011)
25. Maruster, L.: A machine learning approach to understand business processes. Eindhoven University of Technology (2003)
26. Mirkin, B.: Choosing the number of clusters. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **1**, 252–260 (2011)
27. Song, M., Günther, C.W., van der Aalst, W.M.P.: Trace clustering in process mining. In: Ardagna, D., Mecella, M., Yang, J. (eds.) *BPM 2008. LNBIP*, vol. 17, pp. 109–120. Springer, Heidelberg (2009). doi:[10.1007/978-3-642-00328-8_11](https://doi.org/10.1007/978-3-642-00328-8_11)
28. 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)
29. 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)
30. Weidlich, M., Polyvyanyy, A., Desai, N., Mendling, J., Weske, M.: Process compliance analysis based on behavioural profiles. *Inform. Syst.* **36**(7), 1009–1025 (2011)
31. Weijters, A.J.M.M., van der Aalst, W.: Rediscovering workflow models from event-based data using little thumb. *Integr. Comput. Eng.* **10**, 151–162 (2003)

Transactions on Petri Nets and Other Models of
Concurrency XII

Koutny, M.; Kleijn, J.; Penczek, W.; Zhang, M. (Eds.)

2017, XVII, 217 p. 89 illus., Softcover

ISBN: 978-3-662-55861-4