# Clustering Traces using Sequence Alignment

Joerg Evermann[1], Tom Thaler[2,3], and Peter Fettke[2,3]

[1] Memorial University of Newfoundland
[2] Deutsches Forschungszentrum für Künstliche Intelligenz
[3] Universität des Saarlandes

**Abstract.** Process mining discovers process models from even logs. Logs containing heterogeneous sets of traces can lead to complex process models that try to account for very different behaviour in a single model. Trace clustering identifies homogeneous sets of traces within a heterogeneous log and allows for the discovery of multiple, simpler process models. In this paper, we present a trace clustering method based on local alignment of sequences, subsequent multidimensional scaling, and k-means clustering. We describe its implementation and show that its performance compares favourably to state-of-the-art clustering approaches on two evaluation problems.

**Key words:** process mining, process discovery, trace clustering, sequence alignment

## 1 Introduction

Process discovery is that field of process mining that deals with the discovery/mining of process models from event logs. An event log is a set of sequences of events (traces of process instances). Logs may contain traces that differ widely in the sequences of events within them. Mining such heterogeneous event log leads to complicated process models as the mining algorithm constructs models that account for a large proportion of the observed behaviour. Trace clustering addresses this issue by identifying clusters of homogeneous traces within such a heterogeneous log. Constructing a set of process models from sets of homogeneous traces is likely to lead to simpler models.

Most existing trace clustering techniques are based on fairly generic methods for clustering in multivariate settings [1]. These methods operate either by locating individuals in a feature space (e.g. k-means clustering), or directly on a distance[1] matrix between individuals (e.g. k-median clustering) [2].

The main challenge in trace clustering, when using such generic clustering techniques, is the gap between clustering and evaluation [3]. In most existing approaches, the evaluation of process model quality is performed only after clustering is complete. Clustering decisions, e.g. which cluster contains a particular

[1] We use the terms distance and dissimilarity matrix interchangeably, and also use the term similarity matrix synonymously, as one can cluster equally well by maximal similarity or minimal distance.

trace, which cluster to split to combine in hierarchical clustering, are based purely on statistical criteria, i.e. distances in feature space and the inter- and intra-cluster variance, or statistics derived from these. However, depending on how feature space or distance metric are defined, this information is not necessarily a good indicator of the quality of the final process model for each cluster.

There are two ways to address this gap between clustering and evaluation. First, and ideally, mining the model and evaluating its quality should happen during, not after, clustering [3]. However, process model mining and log replay for computing model quality characteristics is computationally expensive. Performing these computations at every step of an iterative clustering algorithm, such as proposed in [3], may be prohibitive for large logs.

Second, one may focus on the definition of an appropriate feature space or distance metric so that it provides the right information for the clustering algorithm to cluster traces that are similar in the sense of being describable by the same simple yet high quality process model. We consider this second issue one of the key challenges in trace clustering. As noted by [2, pg. 506], "specifying an appropriate dissimilarity measure is far more important ... than choice of clustering algorithm. This aspect of the problem is emphasized less in the clustering literature ... since it depends on domain knowledge specifics ...."

In this paper we propose a trace clustering method, called *AlignCluster*, that explicitly takes into account information about the sequences of events in a trace. Specifically, our method uses the Smith-Waterman-Gotoh algorithm for sequence alignment to compute dissimilarity or distances between traces, applies multidimensional scaling to construct a feature space, and then applies k-means clustering. We evaluate our cluster solutions by discovering process models using the flexible heuristics miner (FHM) [4]. The quality of resulting models is typically considered in terms of replay fitness, precision, generalizability and simplicity [5] although information retrieval based measures also exist [6].

The remainder of the paper first introduces our method and then briefly describe its implementation. We then present an evaluation of our method and comparison to state-of-the-art in two evaluation scenarios. Our work is then situated in prior research and the subsequent discussion comprises a brief presentation of future work and some general comments on trace clustering.

## 2 Trace Clustering using Sequence Alignment

Our approach consists of four steps, from preprocessing the log to clustering, described in the following paragraphs.

*Step 1: Preprocessing* We remove duplicate traces while reading the log. This greatly reduces the size of the clustering problem but gives equal "weight" to each unique trace during clustering.

*Step 2: Sequence Alignment* To compare traces, we adopt methods developed in the bio-informatics discipline, which has developed algorithms for optimal

alignment sequences of DNA and protein building blocks. An alignment is a sequence of pairs, either of elements of the two sequences, or of an element of the first and a "gap" in the second sequence, or of a gap in the first and an element of the second sequence. The notion of gaps is similar to "move log" and "move model" operations in log replay techniques for conformance checking [5]. For example, the sequences `GCATGCA` and `GATTACA` may be aligned as ('-' represents a gap):

```
GCATG-CA
G-ATTACA
```

An alignment may be optimal by some scoring system, which consists of the similarity matrix between sequence elements and the gap scoring scheme. For example, a simple scoring system might assign a score of $+1$ for all exact matches, and a score of $-1$ for mismatches and gaps. In general, the similarity matrix between elements depends on the application area and should be defined with substantive knowledge about the sequence elements, i.e. the workflow events.

An early algorithm for optimal alignment was developed by [7] and is a type of *global alignment* algorithm. A variation on this [8] was improved by [9] and is a type of *local alignment* algorithm. The latter type of algorithm identifies multiple regions of smaller optimal alignments and is appropriate when the sequences are of different lengths, as is typically the case with process event logs.

We rely on the Smith-Waterman-Gotoh (SWG) local alignment algorithm [8, 9] for local alignment. Our scoring schema assigns a value of 1 for exact matches and a fixed penalty otherwise (parameter $mismatchPenaltyRelative$, ($mmP$)). Our gap scoring scheme defines a penalty for beginning a gap (parameter $gapOpenCostRelative$ ($gOC$)) and another for extending a gap by one position (parameter $gapExtendCostRelative$ ($gOE$)). These parameters are relative to the value for an exact match.

This step of our method is a critical place to apply substantive business knowledge. For example, the events "customer query processed" and "support request completed" may be highly similar in a particular organization and process, despite the fact that they bear little superficial similarity. While one could try to devise automated comparison of event names, perhaps even based on domain ontologies or WordNet lookup, this can never be a full substitute for application-specific knowledge of the processes that produced the event log.

The SWG algorithm provides two results. The first is the similarity between the aligned sequences as a count of the number of exact matches. The second result is the alignment cost as the sum of penalties for opening and extending gaps in either sequence. Our implementation can use either result to construct the trace similarity or distance matrix. The choice is parametrized using the boolean parameter $useSim$, which, when true, uses the similarity metric, otherwise the cost-based metric is used.

*Step 3: Multi-Dimensional Scaling* Clustering algorithms operate either on features of instances or on a distance matrix. The feature set spans an n-dimensional space in which instances can be located and on which a distance metric can be

defined. Using this metric, computing distances from features is straightforward. Many distance metrics have been defined for numerical characteristics (e.g. euclidean distance, Manhattan distance) but also for character-valued characteristics (e.g. string-edit distances). On the other hand, when only a distance matrix is available, one can use multi-dimensional scaling (MDS) [10] to span a space of arbitrary dimensions and locate the instances in that space. MDS can be considered as an optimization problem:

$$\min_{x_1,\ldots,x_I} \sum_{i<j} (||x_i - x_j|| - \delta_{i,j})^2$$

Here, $||\ldots||$ is the distance metric to be used for the spanned space, $x_i, x_j$ are vectors locating instances $i$ and $j$ in the space, and $\delta$ is the distance between cases $i$ and $j$.

One of the key choices in MDS is the dimensionality of the space. A higher dimensionality allows for a better separation of clusters in the following clustering step. However, when the space gets too sparse, it may be difficult to identify clusters at all. On the other hand, once the space gets too dense because of too few dimensions, clusters may not cleanly separate. Moreover, because the dimensionality of the space is the maximum number of clusters, the dimensionality should not be too small. In our work, the dimensionality of the space is a function of the number of unique traces, e.g. $\sqrt{n}$ and $\log_e n$.

*Step 4: Clustering* One of the key decisions in clustering is the number of clusters to choose. Different approaches to characterizing the quality of a clustering solution and for identifying the optimal number of clusters have been proposed in the literature. However, in the context of trace clustering, these approaches lack a direct connection to the final outcome of the process mining step, i.e. the quality of the resulting process models. This has been termed the clustering versus evaluation bias by [3]. Thus, considerations of within cluster and between cluster sums-of-squares and derived statistics are only of limited value in the context of trace clustering. Hence, rather than investigate the performance of different heuristics for choosing the optimal number of clusters, we defer to the process analyst to evaluate the resulting process models and to make informed decisions about the optimal number of clusters. We use k-means clustering for this research.

## 3 Implementation

We have implemented our approach as a Java application. The application reads a log in CSV format and creates a distance matrix using the SWG algorithm, using the `jaligner` implementation of the SWG algorithm[2]. It then writes the distance matrix and a script for the R statistical system [11] to file and calls the

[2] `http://jaligner.sf.net`

R system to execute the script. MDS and clustering are performed using R and it is easy to substitute different options and parameters at this stage. Our work uses the `cmdscale` function for MDS and the `kmeans` function for clustering. The R script writes a set of files with cluster assignments, which is then read back and used to create logs in XES format.

For evaluation purposes, our application then creates a script for the ProM process mining framework [12] and calls ProM to execute it. ProM reads each XES log and applies the FHM [4] with default parameters. FHM is used in other trace clustering methods as well, specifically the DWS and ActiTraC methods [3, 13] and we found it to be very robust. The heuristics net is then converted to a Petri Net. We then use the log replay technique [14], as implemented in the *PNetReplayer* plugin [15] to compute fitness, and the *PNetAlignmentAnalysis* plugin [16] to compute precision and generalizability.

This scripting approach allows us to automate as much of the process as possible in order to experiment with different values for important parameters, such as the sequence element similarities and gap costs for the SWG algorithm, the dimensionality of the space created by MDS, the number of clusters to identify, and the clustering algorithm. Our implementation is available from the first author's website[3].

## 4 Evaluation

We have evaluated our approach and compared its performance to earlier methods. Specifically, we compared our method to the same set of algorithms as in [1], i.e. Sequence Clustering (SC) [17], Trace Clustering (TC) [18], ActiTraC (AT) [3] and DWS [13]. Implementations for these are provided in the ProM framework [12]. We evaluated our approach in two scenarios, taken from [1].

### 4.1 Evaluation Scenario 1

We constructed a log comprised of 500 traces each from three different logs so that a correct clustering solution exists. Logs from the incident management process at RaboBank Group ICT [19], the loan application process at a Dutch financial institute [20], and the translation process at Leginda.de [21] were randomly extracted and aggregated. Results for the state-of-the-art clustering implementations on separating the three processes in the log, reported by [1], show that only AT was able to cleanly separate the log.

We used the simple scoring scheme described in Sec. 2. As noted earlier, this is the place in our algorithm where substantive business knowledge about the similarity of different activities could be applied. In this artificial problem the number of clusters is known to be 3. The parameter settings in Table 1 are those with which our method was able to cleanly separate the three component logs.

[3] `http://joerg.evermann.ca/software.html`

| | | | |
|---|---|---|---|
| *mismatchPenaltyRelative* | *mmp* | Penalty for an alignment mismatch, relative to an exact match | −1.0 |
| *costGapOpenRelative* | *cGO* | Cost to open a gap, relative to an exact match | 0.0 |
| *costGapExtendRelative* | *cGE* | Cost to extend a gap, relative to the cost of opening a gap | 0.5 |
| *useSim* | | Use similarity, rather than alignment cost | $false$ |
| *numDimensions* | *dim* | Function to compute number of dimensions for MDS from number of traces | *sqrt* |
| *numClusters* | *c* | Number of clusters to identify | 3 |

**Table 1.** Parameters, their descriptions, and settings for evaluation scenario 1

### 4.2 Evaluation Scenario 2

The second scenario is a more realistic case, using a log of 1,500 traces from the above mentioned loan application process [20], available from the first author's website. We evaluated the four quality dimensions of fitness, precision, generalization using existing ProM plugins [15, 16], as described above. For assessing simplicity we refer to [22] and use three different metrics. The cyclomatic number $CN$ is defined as $CN = |A| - |N| + 1$ where $|A|$ is the number of arcs and $|N|$ is the number of nodes in the Petri net. The coefficient of connectivity $CNC$ is defined as $CNC = \frac{|A|}{|N|}$ and the density $\Delta$ is defined as $\Delta = \frac{|A|}{|N|*(|N|-1)}$. We aggregated the quality metrics across clusters using the weighted mean, weighted by the number of traces in each cluster.

Many trace clustering methods are extensively parametrized, as is our own. For our comparison to the state-of-the-art, it is impractical to systematically explore the different parameter settings for existing approaches. We followed [1] and limited the number of configurations. ActiTraC was applied in three configurations (3 clusters; 6 clusters; 6 clusters with ICS set to 0.95). DWS was applied in two configurations (default settings; max clusters per split = max feature length = max splits = 5 and max number of features = 10). Trace clustering was applied in five configurations (default; width 1, height 3; width 2 height 3; width 3, height 3; width 4, height 3). Sequence clustering was applied in five configurations (number of clusters 3, 6, 9, 12, 15). Table 2 shows the performance of the state-of-the-art systems on the different quality dimensions with the best values highlighted.

We conducted an experiment that systematically varied the parameters for our method. Table 3 shows the parameter values we applied, yielding 720 experimental conditions. From these we identified the configuration that yield the optimum outcome for each quality characteristic, shown in the bottom part of Table 2 and plotted in Figure 1. The complete set of 720 results is available from the first author's website.

On this evaluation scenario the optimal configurations were fairly close in performance, even though optimized for different quality criteria (Table 2). For example, generalizability was close to 1 for all configurations, precision close to

| Conf | CNC | CN | Delta | Fit | Prec | Gen | *mmP* | *cGO* | *cGE* | *useSim* | *dim* | *c* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AT-3 | 1.1670 | 33.7120 | 0.0240 | 0.7000 | 0.4737 | 0.7322 | | | | | | |
| AT-6 | 1.1198 | 26.1960 | *0.0326* | *0.6670* | 0.5663 | 0.6011 | | | | | | |
| AT-6-ICS95 | 1.1709 | 27.3087 | 0.0072 | 0.8529 | 0.3751 | 0.9661 | | | | | | |
| DWS-Std | 1.2275 | 30.8013 | 0.0103 | 0.8783 | *0.3219* | 0.9586 | | | | | | |
| DWS-55510 | 1.1579 | 17.3960 | 0.0208 | 0.7721 | 0.5459 | 0.9581 | | | | | | |
| TC | 1.1773 | 31.6533 | 0.0270 | 0.7823 | 0.4062 | 0.7419 | | | | | | |
| TC-W1-H3 | *1.2434* | *46.1867* | 0.0129 | 0.8213 | 0.3232 | 0.8937 | | | | | | |
| TC-W2-H3 | 1.1792 | 35.1787 | 0.0217 | 0.7235 | 0.4413 | 0.8037 | | | | | | |
| TC-W3-H3 | 1.1542 | 31.6587 | 0.0279 | 0.6991 | 0.4686 | *0.7309* | | | | | | |
| TC-W4-H3 | 1.1542 | 31.6587 | 0.0279 | 0.6991 | 0.4840 | 0.7332 | | | | | | |
| SC-3 | 1.1976 | 32.9653 | 0.0071 | 0.8475 | 0.3631 | 0.9598 | | | | | | |
| SC-6 | 1.1346 | 20.5973 | 0.0071 | 0.8644 | 0.5103 | 0.9905 | | | | | | |
| SC-9 | 1.1273 | 17.7667 | 0.0078 | 0.8623 | 0.5408 | 0.9335 | | | | | | |
| SC-12 | 1.1048 | 13.6540 | 0.0083 | 0.8607 | 0.5502 | 0.9628 | | | | | | |
| SC-15 | 1.0974 | 12.0793 | 0.0104 | 0.8919 | 0.5760 | 0.9793 | | | | | | |
| maxPrec | 1.1201 | 15.0000 | 0.0094 | 0.8036 | 0.5992 | 0.9957 | -0.5 | 0.5 | 1.0 | *false* | *log* | 9 |
| maxGen | 1.1507 | 22.0000 | 0.0073 | 0.7775 | 0.5158 | 0.9988 | -0.5 | 1.0 | 0.5 | *false* | *log* | 3 |
| maxFit | 1.1209 | 14.0000 | 0.0104 | 0.8540 | 0.5679 | 0.9965 | -0.5 | 1 | 0.5 | *false* | *log* | 9 |
| minCNC | 1.1164 | 13.0000 | 0.0108 | 0.8502 | 0.5872 | 0.9969 | -1 | 1 | 1 | *false* | *log* | 9 |
| minCN | 1.1164 | 13.0000 | 0.0108 | 0.8502 | 0.5872 | 0.9969 | -1 | 1 | 1 | *false* | *log* | 9 |
| minDelta | 1.1840 | 30.0000 | 0.0072 | 0.7888 | 0.4767 | 0.9941 | -0.5 | 0.5 | 0.5 | *false* | *sqrt* | 3 |

**Table 2.** Performance of the state-of-the-art and different AlignCluster configurations on evaluation scenario 2, best values for each quality criterion shaded, worst values italicized

| | | |
|---|---|---|
| *mismatchPenaltyRelative* | *mmP* | 0.0, −0.5, −1.0, −2.0 |
| *costGapOpenRelative* | *cGO* | 0.0, 0.5, 1.0 |
| *costGapExtendRelative* | *cGE* | 0.0, 0.5, 1.0 |
| *useSim* | | *false*, *true* |
| *numDimensions* | *dim* | *sqrt*, *log* |
| *numClusters* | *c* | 3, 6, 9 |

**Table 3.** Parameters and settings for evaluation scenario 2

0.6 for all but one configuration, and fitness was high at approx. 0.85 even for conditions optimized for simplicity (minCNC, minCN). Examining the parameter values of the configurations shows that while moving from 9 to 3 clusters may have optimized generalizability or minimized $\Delta$, the trade-off on other quality characteristics was significant. The configuration minimizing CNC and CN also performs well on the other quality characteristics. Consequently, we recommend this configuration for logs similar to this log. As this log is similar to others we have encountered, *we recommend this as the default configuration.*

Comparing this default configuration to the performance of existing methods in Table 2 shows that the our method performs comparably to ActiTraC but providing somewhat simpler models with somewhat better precision and generalizability, and comparable trace fitness. It performs comparably to DWS in its standard configuration but with better precision. Our method outperforms Trace Clustering in all its configurations, yielding simpler models with better fitness, precision, and generalizability. Finally, our method performs similar to

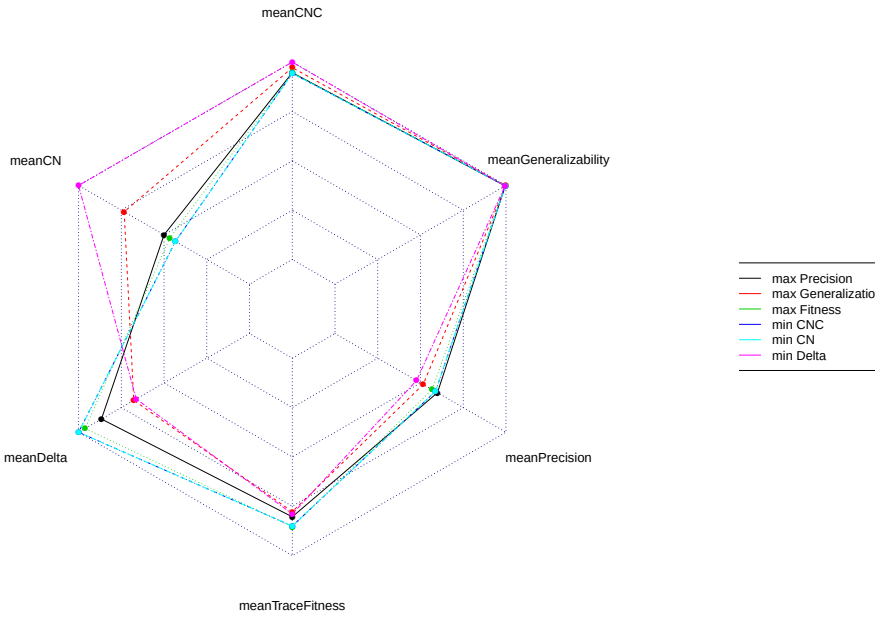


**Fig. 1.** Radarplot of AlignCluster performance on evaluation scenario 2

Sequence Clustering with 9 or more clusters, but has a slight advantage in terms of precision and generalizability.

## 5 Related Work

Sequence alignment and related techniques have been used in process mining before. Most similar to our work is the work on trace alignment by [23, 24]. Also inspired by bio-informatics research, it uses the Needleman-Wunsch algorithm [7] for *global* alignment, rather then *local* alignment as we do. Their work is implemented in the *Trace Alignment* plugin for ProM. However, clustering is not their primary goal and is used only to construct the "guide tree" which guides the selection of sequences for alignment. Clustering in [23, 24] is not based on sequence alignment but is an agglomerative method on a feature space spanned by maximal repeat features [25].

The approach in [26] uses the edit distance between sequences for clustering. While not based on a local or global sequence alignment, as in this paper, string edit distance also accounts for sequence characteristics of traces and is known to be equivalent to sequence alignment approaches [27]. As in our work, models are not mined or evaluated until clustering is completed and the gap between

clustering and evaluation is addressed indirectly through the choice of trace similarity metric.

The model-based approach by [28, 17] represents each cluster by a Markov chain that can generate the behaviour of sequences in that cluster. Traces are assigned to clusters by maximizing the probability that traces are produced by their cluster's Markov chain. An implementation based on commercial tools is described in [28]; an implementation in the *Sequence Clustering* ProM plugin is presented by [17]. This method address the gap between clustering and evaluation directly by using a notion analogous to replay fitness in the clustering algorithm. However, the estimation of Markov Chain parameters is computationally expensive. Our own approach achieves similar, and even slightly better performance, using a less computationally demanding method.

ActiTraC [3] also addresses the gap between clustering and evaluation directly by repeatedly mining and evaluating process models during clustering. It uses maximal repeat features [25] to select candidate traces for assignment to clusters, which are then evaluated using the FHM and replay fitness. Both ActiTraC and Sequence Clustering are computationally intensive. Because it mines process models during clustering, ActiTraC provides not only clusters as a result, but also the generated model, which we have not used in our evaluation. However, as these are also based on the FHM algorithm they are likely to be similar to the ones post-hoc generated in our evaluation.

Other log clustering methods account for sequence information in a more limited way. The *Trace Clustering* method and ProM plugin [18] is based on a feature space of which the "follows" relation between events is the only sequence-related dimension. Additionally, data attributes for cases, event, and performance characteristics are used to span the feature space. A variety of distance metrics and standard clustering methods can then be applied.

DWS [13] is a divisive hierarchical clustering method and ProM plugin that is based on a vector space spanned by frequent features. A frequent feature is a subsequence and a task such that the feature is frequent in the log, but the sequence and appended task are not. By using such sequence frequency information, DWS attempts to include the notion of soundness, which is somewhat analogous to the notion of fitness in that it indicates how well the model explains the traces, into the clustering algorithm.

The method in [29] is based on the number of repeating features (subsequences) in a trace. This accounts for some sequence information, but then abstracts by creating a feature space based on the number of occurrences of subsequences and using these to locate traces in the space for clustering.

## 6 Discussion

The key challenge in trace clustering, bridging the gap between clustering and evaluation, can be addressed in two ways. The direct, but computationally expensive way, exemplified by the ActiTraC [3] and Sequence Clustering [17] methods, incorporates process mining and model evaluation into the clustering method.

We have chosen the less direct, but computationally less demanding way of focusing on the definition of a distance metric and feature space that allows the application of generic multivariate clustering methods. Our experiments, albeit limited in scope to two evaluation scenarios, show that this less direct way is able to yield results as good as those obtained by the more direct approaches.

Despite our initial results, many open questions remain. For example, there is a lack of agreed upon quality characteristics for *sets* of process models. In this paper, we have followed [1] in using simple weighted averages over the clusters, but this is surely a naive view and a better approach rooted in considerations of the process models is required. Consider the optimal results highlighted in Table 2. As the number of clusters increases, the models tend to improve in quality. This reduction in model size, and improvement in fitness and precision is expected as it is the main reason for performing clustering in the first place. Hence, what is required is either a "correction" or "adjustment" for the number of clusters (models) or a restriction to clustering solutions with the same number of clusters.

Second, there may be possible interactions between the clustering method and the mining method. For example, a particular mining algorithm might yield better results when operating on clusters from one clustering method than from another clustering method. For example, a mining algorithm geared towards discovering parallelism requires clusters that separate choices, while an algorithm that favors choice requires clusters with sequential parallel behaviour[4]. Despite some concerns about the use of heuristics and the lack of semantic preserving translation from heursistic models to Petri nets, in this work we have used only the FHM as it is also used by other approaches and yields sensible models in many situations. Exploring this potential inter-dependency remains a challenge for future research.

There are many possible extensions to the method presented here. First, the elimination of duplicate traces in the log greatly reduces the size of the problem, but may bias the resulting quality metrics, computed by replaying each trace against a model. Consequently, a variation of our method may choose to not remove the duplicate traces and opt for worse runtime performance instead.

We use a *local* alignment method, whereas [23, 24] use a *global* alignment method. The latter is better suited to sequences that are similar in length. It may be worthwhile to compare the performance of these types of alignment methods and choose based on log characteristics.

An optimal alignment offers a range of possible similarity or distance measures beyond the simple match/mismatch and gap-cost-based ones explored here. One possibility is to use a weighted mean of the two measures we have presented. Another possibility is to not simply count the matches and mismatches, but to weight these by the similarity of the trace events that are aligned.

Finally, a variation of k-means clustering is k-median clustering. This uses the distance matrix for clustering and removes the need to use MDS for constructing. In k-median clustering the cluster center is represented by a specific individual.

[4] We thank one of the anonymous reviewers for this specific example.

Given the typically large number of traces in trace clustering applications, we expect differences to k-mean clustering to be minimal, as there is likely to be an individual very close to the k-mean cluster center.

In conclusion, this initial application of sequence alignment to trace clustering shows promising results and improves on the existing state-of-the-art, but much future research remains in this area.

## References


1. Thaler, T., Ternis, S.F., Fettke, P., Loos, P.: A comparative analysis of process instance cluster techniques. In Thomas, O., Teuteberg, F., eds.: Smart Enterprise Engineering: 12. Internationale Tagung Wirtschaftsinformatik, WI 2015, Osnabrück, Germany, March 4-6, 2015. (2015) 423–437
2. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer (2009)
3. De Weerdt, J., Vanthienen, J., Baesens, B., et al.: Active trace clustering for improved process discovery. Knowledge and Data Engineering, IEEE Transactions on **25**(12) (2013) 2708–2720
4. Weijters, A., Ribeiro, J.: Flexible heuristics miner (FHM). In: Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining CIDM 2011, Paris, France. (2011)
5. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer Verlag, Heidelberg, Germany (2011)
6. De Weerdt, J., De Backer, M., Vanthienen, J., Baesens, B.: A robust f-measure for evaluating discovered process models. In: Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2011, part of the IEEE Symposium Series on Computational Intelligence 2011, April 11-15, 2011, Paris, France, IEEE (2011) 148–155
7. Needleman, S.B., Wunsch, C.D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology **48**(3) (1970) 443 – 453
8. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. Journal of molecular biology **147**(1) (1981) 195–197
9. Gotoh, O.: An improved algorithm for matching biological sequences. Journal of molecular biology **162**(3) (1982) 705–708
10. Cox, T.F., Cox, M.A.: Multidimensional scaling. CRC Press (2000)
11. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. (2014)
12. van Dongen, B.F., de Medeiros, A.K.A., Verbeek, H.M.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: The ProM framework: A new era in process mining tool support. In: Proceedings of the 26th International Conference on Applications and Theory of Petri Nets, Berlin, Heidelberg, Springer-Verlag (2005) 444–454
13. De Medeiros, A.K.A., Guzzo, A., Greco, G., Van Der Aalst, W.M., Weijters, A., Van Dongen, B.F., Saccà, D.: Process mining based on clustering: A quest for precision. In: Business Process Management Workshops, Springer (2008) 17–29
14. Rozinat, A., van der Aalst, W.M.: Conformance checking of processes based on monitoring real behavior. Information Systems **33**(1) (2008) 64–95

15. van der Aalst, W.M.P., Adriansyah, A., van Dongen, B.F.: Replaying history on process models for conformance checking and performance analysis. Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery **2**(2) (2012) 182–192
16. Adriansyah, A., Munoz-Gama, J., Carmona, J., van Dongen, B.F., van der Aalst, W.M.P.: Alignment based precision checking. In Rosa, M.L., Soffer, P., eds.: Business Process Management Workshops - BPM 2012 International Workshops, Tallinn, Estonia, September 3, 2012. Revised Papers. Volume 132 of Lecture Notes in Business Information Processing., Springer (2012) 137–149
17. Veiga, G.M., Ferreira, D.R.: Understanding spaghetti models with sequence clustering for ProM. In: BPM Workshops, Springer (2010) 92–103
18. Song, M., Günther, C.W., Van der Aalst, W.M.: Trace clustering in process mining. In: Business Process Management Workshops, Springer (2009) 109–120
19. Van Dongen, B., Weber, B., Ferreira, D., De Weerdt, J.: Business process intelligence challenge (BPIC'14) (2014)
20. Van Dongen, B., Weber, B., Ferreira, D.: Business process intelligence challenge (BPIC'12) (2012)
21. Thaler, T., Fettke, P., Loos, P.: Process mining - Fallstudie leginda.de. HMD Praxis der Wirtschaftsinformatik **293** (2013) 56—66
22. Melcher, J.: Process Measurement in Business Process Management – Theoretical Framework and Analysis of Several Aspects. KIT Scientific Publishing, Karlsruhe, Germany (2012)
23. Jagadeesh Chandra Bose, R.P., van der Aalst, W.M.P.: Process diagnostics using trace alignment: Opportunities, issues, and challenges. Inf. Syst. **37**(2) 117–141
24. Jagadeesh Chandra Bose, R., van der Aalst, W.: Trace alignment in process mining: Opportunities for process diagnostics. In Hull, R., Mendling, J., Tai, S., eds.: Business Process Management. Volume 6336 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2010) 227–242
25. Bose, R.J.C., van der Aalst, W.M.: Trace clustering based on conserved patterns: Towards achieving better process models. In: Business Process Management Workshops, Springer (2010) 170–181
26. Bose, R.P.J.C., van der Aalst, W.M.P.: Context aware trace clustering: Towards improving process mining results. In: Proceedings of the SIAM International Conference on Data Mining, SDM 2009, April 30 - May 2, 2009, Sparks, Nevada, USA, SIAM (2009) 401–412
27. Sellers, P.H.: On the theory and computation of evolutionary distances. SIAM Journal on Applied Mathematics **26**(4) (1974) 787–793
28. Ferreira, D.R.: Applied sequence clustering techniques for process mining. In Cardoso, J., van der Aalst, W., eds.: Handbook of Research on Business Process Modeling. Information Science Reference, Hershey, PA (2009) 481–502
29. Bose, R., van der Aalst, W.: Trace clustering based on conserved patterns: Towards achieving better process models. In Rinderle-Ma, S., Sadiq, S., Leymann, F., eds.: BPM Workshops. Springer Berlin Heidelberg (2010) 170–181