Active Trace Clustering for Improved Process Discovery

Jochen De Weerdt, Seppe vanden Broucke, Jan Vanthienen, and Bart Baesens

Abstract—Process discovery is the learning task that entails the construction of process models from event logs of information systems. Typically, these event logs are large data sets that contain the process executions by registering what activity has taken place at a certain moment in time. By far the most arduous challenge for process discovery algorithms consists of tackling the problem of accurate and comprehensible knowledge discovery from highly flexible environments. Event logs from such flexible systems often contain a large variety of process executions which makes the application of process mining most interesting. However, simply applying existing process discovery techniques will often yield highly incomprehensible process models because of their inaccuracy and complexity. With respect to resolving this problem, trace clustering is one very interesting approach since it allows to split up an existing event log so as to facilitate the knowledge discovery process. In this paper, we propose a novel trace clustering technique that significantly differs from previous approaches. Above all, it starts from the observation that currently available techniques suffer from a large divergence between the clustering bias and the evaluation bias. By employing an active learning inspired approach, this bias divergence is solved. In an assessment using four complex, real-life event logs, it is shown that our technique significantly outperforms currently available trace clustering techniques.

Index Terms—Process mining, trace clustering, active learning, event logs

1 Introduction

Process mining has been demonstrated to possess the capabilities to profoundly assess business processes [1]. In particular, process mining techniques are highly suitable in flexible environments such as healthcare, customer relationship management (CRM), product development and so on [2]. This is because information systems in such environments often grant a higher degree of freedom to their users. Accordingly, process mining proves to be valuable by discovering the actual process at hand.

The starting point of analysis is an *event log*, which is basically a set of process executions capturing the different business activities that were performed in the context of a certain case. In this study, the control-flow perspective, i.e., the different transition relationships between the activities in the event log, is the object of analysis. However, typical event logs will contain much more information, for instance organizational information concerning the performers of the different activities [3].

The most crucial learning task in the process mining domain is termed *process discovery* and is defined as the construction of a process model from an event log [4], [5]. Process discovery is a largely unsupervised learning task in nature due to the fact that event logs rarely contain negative events to record that a particular activity could not have taken place. Despite the demonstrated usefulness in flexible environments, it has been shown that *process discovery* is most challenging in this context. Various studies illustrate that process discovery techniques experience difficulties to render accurate and interpretable process models out of event logs stemming from highly flexible environments [2], [6], [7], [8]. This problem is largely due to the high variety of behavior that is captured in certain event logs. Accordingly, different approaches have been proposed to cope with this issue. Next to event log filtering, event log transformation [9] and tailor-made discovery techniques such as Fuzzy Mining [10], *trace clustering* can be considered a versatile solution for reducing the complexity of the learning task at hand. This is because one can rely on multiple process models to represent the variety in behavior of a certain event log by separating execution traces into different groups. The purpose of clustering process executions is detailed in Fig. 1. By dividing traces into different groups, process discovery techniques can be applied on subsets of behavior and thus improve the accuracy and comprehensibility.

In this study, we build further on the idea of clustering event log traces to reduce the complexity of the process discovery learning task. Therefore, we describe a novel approach that aims at improving currently available techniques by directly optimizing the accuracy of the cluster's underlying process models. In this way, the gap between the clustering bias and the evaluation bias from which currently available techniques suffer, is bridged. As such, this paper is structured as follows: In Section 2, we provide an overview of existing approaches to trace clustering. Section 3 details the new trace clustering approach. Then Section 4 introduces multiple metrics for evaluating the quality of a trace clustering technique. These metrics are employed in Section 5

- *J. De Weerdt, S. vanden Broucke, and J. Vanthienen are with the Department of Decision Sciences and Information Management, Katholieke Universiteit Leuven, Naamsestraat 69, B-3000 Leuven, Belgium. E-mail: {jochen.deweerdt, seppe.vandenbroucke, jan.vanthienen}@kuleuven.be.*
- *B. Baesens is with the Department of Decision Sciences and Information Management, Katholieke Universiteit Leuven, Naamsestraat 69, B-3000 Leuven, Belgium and with the School of Management, University of Southampton, Highfield Southampton, SO17 1BJ, United Kingdom. E-mail: bart.baesens@kuleuven.be.*

Manuscript received 23 July 2012; revised 11 Oct. 2012; accepted 7 Nov. 2012; published online 16 Nov. 2012.
Recommended for acceptance by N. Mamoulis.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2012-07-0526.
Digital Object Identifier no. 10.1109/TKDE.2013.64.

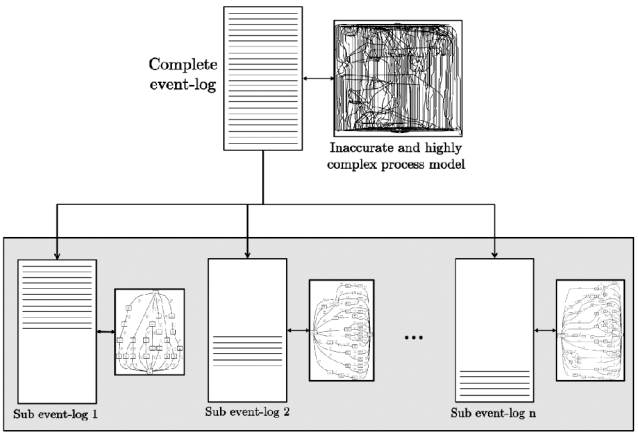


Fig. 1. Illustration of the purpose of trace clustering in process mining.

in which the novel trace clustering technique is assessed and compared to existing techniques, both in a controlled environment as well as by making use of four real-life event logs. Finally, Section 6 provides a discussion before conclusions are formulated in Section 7.

2 Related Work

In the literature, different approaches to trace clustering have been proposed. Many techniques apply a kind of translation to the learning problem so as to make use of the existing distance-based clustering algorithms. For instance, by converting an event log into a vector space model, a distance metric can be defined between each couple of traces. However, there exist techniques that define the distance between two traces without the translation to an attribute-value context. Furthermore, model-based clustering techniques were shown to be applicable for trace clustering as well.

Vector space approaches. Greco et al. [11] were pioneers in studying the clustering of log traces within the process mining domain. They use a vector space model considering the activities and the activity transitions to cluster the traces in an event log with the purpose of discovering more simple process models for the subgroups. The authors propose the use of disjunctive workflow schemas (DWS) for discovering process models. The underlying clustering methodology is k-means clustering.

Song et al. [12] elaborate on the idea of constructing a vector space model for traces in an event log. In contrast to [11], this technique allows for a multitude of so-called *profiles* to determine the vector associated with each process instance. As such, they define activity, transition, performance, case attribute profiles and so on. Furthermore, the implementation of the technique presents a full range of distance metrics and clustering techniques.

Context-aware trace clustering. The most recent trace clustering techniques in process mining are described by Jagadeesh Chandra Bose and van der Aalst [7], [13]. They extend contemporary approaches by improving the way in which control-flow context information is taken into account. Please note that context-aware refers to control-flow properties of the traces in the event log and not to *contextual* information such as originators, case data, and so on. In [7], Jagadeesh Chandra Bose and van der Aalst propose a generic edit distance technique which is founded on the Levenshtein distance [14]. The approach relies on deriving specific substitution, insertion, and deletion costs so as to take into account the behavior in the event log. The idea of context-aware trace clustering is further developed in [13] in which the authors return to the principle of generating a vector space model for the traces in an event log. However, instead of using activities and/or transitions as a basis for the vector space model, it is proposed to use conserved patterns or subsequences. In this way, the authors define maximal, super maximal, and near super maximal repeats to create feature sets that determine the vector of a certain trace.

Model-based sequence clustering. A totally different approach to trace clustering was proposed by Ferreira et al. [15]. Inspired by the work of Cadez et al. [16] in the area of web usage mining, they propose to cluster sequences by learning a mixture of first-order Markov models using the expectation-maximization (EM) algorithm. In [8], this model-based trace clustering technique is applied to server logs with the purpose of demonstrating its usefulness in a real-life setting. Note that the technique presented in the next section is most similar to [15], notwithstanding that the underlying model representation and the clustering strategy is different.

3 Active Trace Clustering

This section introduces the new trace clustering approach. Its conception is significantly different from earlier techniques because it starts from the observation that traditional trace clustering techniques suffer from a divergence between the clustering bias and the evaluation bias.

3.1 Clustering Bias versus Evaluation Bias

The key idea behind existing trace clustering algorithms is to group traces that exhibit similar features. Although this idea is intuitive and has proven useful, it suffers from a fundamental problem. In a traditional data mining setting, clusters are created and evaluated based upon the idea of maximizing intra-cluster similarity and minimizing inter-cluster similarity. As such, currently available clustering techniques apply either a hierarchical or a partitional method like k-means for creating clusters. However, as described in [13], the most important evaluation dimension for trace clustering is from a process discovery perspective. This entails that for each cluster, a certain discovery technique will create a process model and the corresponding process model accuracy and process model complexity are combined to determine the quality of a certain clustering solution. Because existing techniques do not take into account the quality of the underlying process models of each of the clusters during the clustering procedure, these techniques suffer from a strong divergence between the evaluation bias and the clustering bias. Accordingly, it is highly questionable whether the existing clustering approaches will yield satisfactory results in terms of process models. In Section 5, it will be shown that this bias divergence indeed causes inferior results for existing trace clustering approaches.

Against this background, this paper puts forward an entirely different approach to trace clustering. It does not rely on a vector space model, nor does it define a metric for quantifying similarity between two process instances. Instead, the technique proposed in this study is designed so as to solve the problem of finding an optimal distribution

of execution traces over a given number of clusters whereby the combined accuracy of the associated process models is maximized. Note that brute-force solution strategies for this optimization problem are computationally intractable because they scale exponentially in terms of the number of distinct process instances in the log. For instance, in case of only 10 distinct traces and four clusters, more than 1 million different solutions are possible. Therefore, our technique proposes a top-down, greedy algorithm that computes a solution for this problem by grouping traces not because they exhibit similar behavior but because they fit well in a certain process model.

3.2 An Active Learning Inspired Approach

We term the novel clustering technique *active trace clustering (ActiTraC)*. It is called *active* since it is inspired by a couple of principles of active learning in the field of machine learning. For a detailed survey of active learning literature, we refer to Settles [17]. The key idea behind active learning is that a machine learning algorithm can achieve better results with fewer training data if it is allowed to choose the data from which it learns [18], [19]. Despite the fact that active learning is generally employed in the context of supervised learning techniques to reduce the classification error or label uncertainty (e.g., [20]), studies like [21] have demonstrated that the principles of active learning can be employed in an *unsupervised* setting as well. In this way, our proposed trace clustering technique borrows a couple of elements from the general idea of active learning. Most importantly, a *selective sampling* strategy is employed. Typically, an active learner will decide upon which instances to select based on some informativeness measure. In this case, the frequency of a trace is used as the primordial measure for its informativeness. Furthermore, the proposed technique is a greedy approach which does not guarantee an optimal solution in terms of process model accuracy. Finally, the clustering solution will often be based on a subset of the event log traces. Nonetheless, it is recognized that there exist differences as well. For instance, active learning in a supervised context focuses on problem areas for which the current model is uncertain. Mostly due to the fact that our approach is situated in an unsupervised context, our technique does not really focus on problem areas.

3.3 Notation

Before outlining the ActiTraC algorithm, some important concepts and notations that are used in the remainder of this study are discussed. An event log (L) consists of events (*ev*) that pertain to process instances. A process instance (*pi*) is defined as a logical grouping of activities whose state changes are recorded as events. As such, it is assumed that it is possible to record events such that each event refers to an activity type (*at*), i.e., a step in a process instance, a case, i.e., the process instance, and that events are totally ordered. Note also that identical process instances (i.e., traces with the same sequence of events) can be grouped into a distinct process instance, further denotes as *dpi*. A *dpi* is defined as a set of *pi* with the number of *pi* in the set denoted as the frequency of the *dpi*. A collection of *dpi*'s is called a grouped event log (GL).

3.4 A Three-Phase Algorithm

In general, active trace clustering aims at creating clusters of event log traces for which the resulting process model is accurate according to some metric. The algorithm consists of three distinct phases: *selection*, *look ahead*, and *residual trace resolution*. Its pseudocode is depicted in Algorithm 1. Before the actual three phases of the algorithm can be carried out, identical process instances pi in event log L are grouped into distinct process instances *dpi*. Furthermore, the set of created clusters CS is initialized to the empty set and the set of remaining distinct process instances R contains all *dpi*'s created in the first step.

Algorithm 1 ActiTraC

Input: An event log L, the number of clusters nb_{clus}, a target fitness *tf*, a minimum cluster size *mcs*, a window size w, and a boolean N that is true in case a separate cluster should be created for the remaining traces
Output: A collection of event logs, represented by a set of clusters CS

Convert L into a grouped event log GL
$CS \leftarrow \emptyset$ # CS *denotes the set of clusters*
$R \leftarrow GL$ # R *denotes the set of remaining dpi's*
while $(|CS| < nb_{clus}) \wedge (R \neq \emptyset)$ **do**
  $C \leftarrow \emptyset$ # C *denotes the set of dpi's in the cluster*
  $I \leftarrow \emptyset$ # I *denotes the set of ignored dpi's*

Phase 1: Selection

  repeat
    Define W as the union of the most frequent *dpi* in $R \setminus I$ and the set of the top w% *dpi*'s in $R \setminus I$ according to frequency # w *specifies the size of the window*

    if $C = \emptyset \vee |W| = 1$ **then**
      $cur\_dpi \leftarrow \arg\max_{dpi \in R} |dpi|$
      # *cur_dpi* denotes the *dpi* from R in consideration for addition to the current cluster

    else
      $cur\_dpi \leftarrow \arg\min_{dpi \in W} (dist_{\text{MRA}}(C, dpi))$
      # *function* dist_{MRA} *is the* ***average*** *MRA-based Eucl. distance between the dpi's in C and cur_dpi*
    end if
    $PM \leftarrow HM(C \cup \{cur\_dpi\})$
    if $fitness(PM) \geq tf$ **then**
      $C \leftarrow C \cup \{cur\_dpi\}$
      $R \leftarrow R \setminus \{cur\_dpi\}$
    else
      if $\sum_{dpi \in C} |dpi| \geq mcs \times \sum_{dpi \in R} |dpi|$ **then**
        $PM \leftarrow HM(C)$

        Phase 2: Look ahead

        for all $dpi \in R$ **do**
          if $fits(dpi, PM)$ **then**
            $C \leftarrow C \cup \{dpi\}$
            $R \leftarrow R \setminus \{dpi\}$
          end if
        end for
        exit repeat

      else
        $I \leftarrow I \cup \{cur\_dpi\}$
      end if
    end if
  until $(R = \emptyset) \vee (R = I)$
  add the constructed cluster C to CS
end while

Phase 3: Residual trace resolution

if N **then**
  add new cluster C to CS with $C \leftarrow R$
else
  add each *dpi* in R to that cluster in CS for which its fitness, as calculated on the underlying process model, is highest
end if

3.4.1 Selection

In the first phase, traces are iteratively selected using a selective sampling strategy. The goal is to add a new distinct process instance to the set of already selected instances, with the purpose of evaluating the process model discovered from this new subevent log. If the process model remains accurate enough, the selected trace is added to the current cluster and the selection procedure is repeated. As long as it is possible to add a trace to the current cluster without decreasing the process model accuracy too much w.r.t. a given *target fitness (tf)*, this process continues. To deal with the problem that small clusters are created, the *minimum cluster size (mcs)* parameter, ranging between 0 and 1, can be used so as to continue the selection and model building phase when an instance is encountered that results in an unfit process model (line 20). In this way, the algorithm can skip certain traces to increase the size of the current cluster.

Frequency-based selective sampling. The *window size (w)* parameter in line 9 gives rise to two variants of the ActiTraC algorithm. In case w is set to 0, the frequency window W will only contain the most frequent *dpi* in R. In this case, a straightforward selective sampling strategy is put in place that beholds the selection of the most frequent *dpi*. For this selected *dpi* and all the *dpi*'s already in C (with C denoting the set of *dpi*'s being part of the current cluster), a process model (*PM*) is calculated based on the function *HM*. *HM* denotes the application of the HeuristicsMiner algorithm [22] to the set of *dpi*'s. In the remaining part, this frequency-based selective sampling algorithm will be denoted $\mathrm{ActiTraC}^{\mathrm{freq}}$.

Distance-based selective sampling. Next to the basic frequency-based sampling, ActiTraC provides the flexibility to include a more complex selection strategy. In fact, ActiTraC can be adapted so as to take into account any kind of distance function between distinct process instances. As such, by increasing the window size w, it becomes possible to define a selective sampling strategy that incorporates the idea of clustering more similar traces together. In this study, it is opted to include an MRA-based (maximal repeat alphabet) euclidean distance function as defined in [13]. This ActiTraC variant is further denoted as $\mathrm{ActiTraC}^{\mathrm{MRA}}$. The MRA-technique was selected because of its computational tractability and because it was found better performing in [13]. Note that the distance function in line 13 of Algorithm 1 is applied as follows: from the current set of remaining *dpi*'s, a window is defined based on frequency. For example, the 25 percent most frequent *dpi* form the window. From the set of *dpi*'s that form the window, that *dpi* is selected exhibiting the lowest *average* MRA-based euclidean distance with respect to the current set of *dpi*'s in C. In this way, more similar traces in terms of control-flow behavior are tested first and the resulting clustering solution will significantly differ from the first approach solely relying on frequency. Note that it was opted to rely on frequency partly by defining a window to guide the clustering process towards creating a sufficiently large cluster with respect to the amount of remaining traces. In a similar way as with frequency-based selective sampling, the *dpi* is added to the current cluster and a process model (*PM*) is calculated with HeuristicsMiner.

Once this process model is created, it is verified whether the fitness of the process model is still above a predefined threshold, the *target fitness (tf)*. If the fitness remains higher than the *tf*, the *dpi* is added to the current set of instances C and a new selection is initiated. However, if the process model fitness drops below *tf*, it should be verified whether the *minimum cluster size (mcs)* is attained. If so, selection is halted and phase two commences. Nevertheless, if the *mcs* is not met, the currently selected *dpi* is added to a set of skipped *dpi*'s I and the selective sampling of a new trace from R continues.

3.4.2 Look Ahead

Once a process instance is encountered that decreases the model's accuracy below the specified threshold of the current cluster and the minimum cluster size is reached, the first phase comes to an end. In the second, *look ahead* phase, the remaining instances in the event log are taken into consideration (i.e., those traces that have not been subject to the selection phase yet) by verifying whether some of these traces do fit the process model created in the first phase. In this forward-looking procedure, only distinct process instances that fit the current process model perfectly (i.e., instances with an individual fitness of 1), are added to the current cluster. Instances that do not fit the model remain in the event log for which the selection phase can be started again to create a second cluster. This iteration of selection and look ahead is continued until the predefined maximum number of clusters is reached.

3.4.3 Residual Trace Resolution

Finally, the third phase specifies the resolution of the remaining traces in the event log. Either the remaining instances can be separated into a distinct cluster or these traces can be distributed over the created clusters according to the individual trace fitness for the different process models created. Note that in the remainder, only the latter option will be investigated because this choice will prevent the creation of strongly skewed clusters in terms of their size, which might in its turn trick the evaluation criteria as presented in Section 4.

3.5 Assumptions

The presented algorithms can deal with most event logs. There exists one major assumption regarding the frequency distribution of the process executions. One of the underlying principles is to prioritize traces with higher frequency over traces with a lower frequency. Of course, this can only be achieved in case of a nonuniform distribution of trace frequencies. It can be assumed that in a majority of real-life event logs, a nonuniform distribution is observed. However, it is pointed out that for event logs with a close to uniform distribution, the algorithm will present a result, but it will be challenging to attain sufficiently large clusters in the first two phases of the algorithm.

Further, the algorithm is also subject to the implementation of a mining technique and an accuracy evaluation measure. Currently, it was opted to use HeuristicsMiner [22] as the underlying discovery technique. This is because HeuristicsMiner has been demonstrated to possess the best capabilities to deal with real-life event logs [23]. The low

computational costs of HeuristicsMiner as compared to other available techniques is an essential requirement to keep control over the computational demands of ActiTraC. Finally, ActiTraC relies on the ICS-fitness [24] for both the calculation of the overall accuracy of a process model as well as for the calculation of the individual fitness of a single process instance. This metric is similar to the well-known fitness metric defined by Rozinat and van der Aalst [25], even though it is a metric for heuristic nets. Note that next to the scalability of the metric's replay procedure, due to the avoidance of costly state space calculations, another advantage of ICS-fitness is that it is not a pure recall metric since it also punishes overly general models.

4 Cluster Quality Criteria

The evaluation of a trace clustering solution can be approached from two perspectives. On the one hand, trace clusters can be assessed based on whether they group traces sharing domain-specific characteristics. On the other hand, a set of trace clusters can be evaluated from a process mining perspective, judging whether the technique achieves the goal of creating more accurate and more comprehensible process models. In the next sections, both evaluation dimensions will be detailed.

4.1 Domain-Based Evaluation

Quantifying the quality of a trace clustering technique from a domain perspective suffers from an important drawback which originates in the unsupervised nature of the learning problem. Without any ex ante information about how many domains should be found and which execution traces belong to which domain, it is unfeasible to construct domain-based evaluation metrics. In case there is ex ante information available, for instance in a gold standard evaluation setting, it is possible to define evaluation metrics that quantify how well a certain technique distinguishes the different classes of behavior. Examples of such metrics are true positives, false positives, purity and so on. Another powerful metric that can quantify clustering quality in a supervised setting is entropy. Entropy, a measure of disorder, has its origins in information theory, but can be applied advantageously in a clustering setting. For the purpose of this study, the entropy of a set of trace clusters is defined as follows:

Cluster set entropy (H_{CS}). Let k denote the number of clusters, q the number of classes, n the number of *pi*'s in the event log, n_i the number of *pi*'s in cluster i, and n_{ij} the number of *pi*'s in cluster i belonging to class j. Then H_{CS} is defined as follows:

$$H_{CS} = -\sum_{i=1}^{k}\left(\frac{n_i}{n}\sum_{j=1}^{q}\left(\frac{n_{ij}}{n_i} log_2 \frac{n_{ij}}{n_i}\right)\right). \quad (1)$$

4.2 Process Mining-Based Evaluation

Next to domain relevancy, an even more crucial evaluation perspective is from a process mining viewpoint. With the goal of any trace clustering technique being to improve the accuracy and complexity of the process models discovered from an event log, both accuracy and complexity should be included in a proper quantification of the quality of a trace clustering solution. Note that this evaluation approach is largely inspired by the ideas put forward in [13].

4.2.1 Process Model Accuracy

Process model accuracy quantifies how well the observed behavior in an event log is captured by the discovered process model. In the process mining literature, assessing the quality of a process model with respect to a corresponding event log is termed conformance checking [25], with *trace replay* an often employed technique to calculate metrics. In [26], we have argued that assessing the accuracy of a discovered process model ideally involves an evaluation of multiple dimensions, of which recall and precision are to be considered most important.

In [27], the application of the F-score was shown to be useful to assess the overall accuracy of a process model. This F-score weighs state-of-the-art recall and precision metrics, which are founded on a technique to induce artificial negative events into an event log [28]. For a formal definition and more details about the calculation method of the F-score, we refer to [27].

Obviously, individual F-scores for each of the clusters should be aggregated to assess a clustering solution as a whole. To take into account differences in size between the clusters, it was opted to weigh the individual F-scores based on the number of traces (*pi*) in each cluster. Note that the F-score also allows to weigh recall and precision differently. In this case, it was opted to weigh them equally, which is denoted as the F1-score. As such, the weighted average F1-score is defined as follows.

Weighted average F1-score ($\mathrm{F1}_{W.A.}$). Let n_i denote the number of traces in cluster i $(1 \leq i \leq k)$, $\mathrm{F1}_i$ is the F1-score of the process model for trace cluster i. Then $\mathrm{F1}_{W.A.}$ is defined as follows:

$$\mathrm{F1}_{W.A.} = \frac{\sum_{i=1}^{k} n_i \mathrm{F1}_i}{\sum_{i=1}^{k} n_i}. \quad (2)$$

Note that due to the incorporation of both recall and precision, the weighted average F1-score will punish process models that do not represent the behavior in the event log very well, as well as process models which overgeneralize the behavior.

4.2.2 Process Model Complexity

Next to improving process model accuracy, the goal of trace clustering is to render more comprehensible process models. This comprehensibility can be quantified using process model complexity metrics.

In the literature, Lassen and Van der Aalst [29] describe a number of metrics for quantifying process model complexity: extended Cardoso metric (ECaM), extended Cyclomatic metric (ECyM), and structuredness metric (SM). The main issue with these metrics is that they are designed from a process modeling perspective and thus assume save and sound workflow nets (WF-nets). Since process discovery techniques cannot guarantee the discovery of nets that comply with these properties, it was opted to define a more straightforward metric.

The definition of this metric is supported by the work of Mendling et al. [30] who perform a study on the influence factors of process model understandability. A main result of this study is that very simple metrics such as the number of elements, the number of arcs, and so on are relatively good quantifiers of process model complexity. Based on the assumption that the process models underlying the formed clusters are represented by Petri nets [31], we define the place/transition connection degree as follows.

Place/transition connection degree (P/T-CD). Let $|a|$ represent the number of arcs in a Petri net, $|P|$ the number of places and $|T|$ the number of transitions. Place/transition connection degree P/T-CD is defined as follows:

$$\text{P/T-CD} = \frac{1}{2}\frac{|a|}{|P|} + \frac{1}{2}\frac{|a|}{|T|}. \quad (3)$$

In other words, P/T-CD is the weighted sum of the average number of arcs per transition and the average number of arcs per place. It is argued that the complexity of each of the models adds equally toward the difficulty of interpreting the entire clustering solution and therefore it is opted to not weigh the model complexity according to the sizes of the cluster. Accordingly, the average place/transition connection degree is defined as follows.

Average place/transition connection degree (P/T-CD_A). Let k denote the number of clusters in the clustering solution, and P/T-CD_i the place/transition connection degree of the process model of cluster i. Then P/T-CD_A is defined as follows:

$$\text{P/T-CD}_A = \frac{\sum_{i=1}^{k} \text{P/T-CD}_i}{k}. \quad (4)$$

It is argued that this connection degree metric is an appropriate quantifier of process model complexity, especially for the purpose of evaluating trace clustering techniques because incomprehensible process models are often characterized by a drastic increase in the number of arcs that connect places and transitions.

5 Experimental Evaluation

In this section, the techniques proposed in this study are evaluated in two different settings. First of all, a controlled environment is devised in which the parameter configurations of the ActiTraC-algorithms are analyzed. Furthermore, this supervised experiment is used to assess the capabilities of ActiTraC and existing techniques of identifying ex ante defined classes of behavior. The remainder of this section describes a scalability assessment of ActiTraC and a benchmarking study based on four real-life event logs. The trace clustering techniques presented in this paper are implemented as the *ActiTraC plugin* in ProM 6,[1] an academic process mining framework which allows the use and development of a large variety of techniques for analyzing event logs.

5.1 Controlled Environment Evaluation

Because there does not exist a comprehensive benchmarking environment for trace clustering techniques in the process mining literature, it is opted to design a dedicated controlled evaluation environment. Such an experimental evaluation allows for assessing whether the trace clustering techniques are capable of identifying ex ante defined classes of behavior.

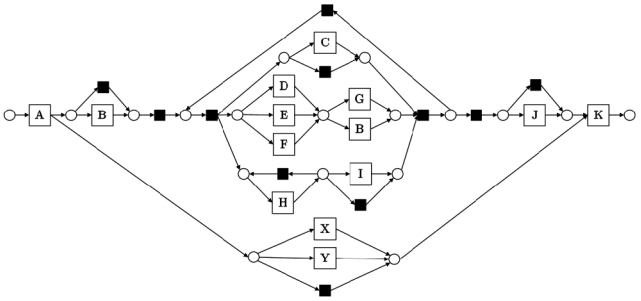


Fig. 2. Petri net used for generating different classes of behavior according to the presence/absence of activities X and Y.

5.1.1 Design

A first essential design choice consists of the definition of different classes of behavior. Therefore, traces have to be generated so as to adhere to a certain class of behavior as determined by a *control-flow based* domain criterion. Such criteria can be very diverse, for example, presence or absence of activities, presence or absence of combinations of activities, looping behavior and so on. In the context of this assessment, it was decided to opt for a straightforward criterion, namely the presence or absence of a single activity. In this way, the Petri net in Fig. 2 was designed to generate different event logs with three classes of behavior: traces where activity X is present, traces where activity Y is present and traces where neither activity X nor activity Y is present. Note that it was opted to include plenty of parallel and looping behavior to approach the complexity of a real-life event log.

Using this Petri net, three different event logs were generated randomly using CPN Tools,[2] containing 2,000 cases of each behavioral class (6,000 cases in total) and with each behavioral class accounting for 200 distinct process instances. The three event logs differ in terms of the distribution of distinct process instances. The exponential (linear) log presents an exponentially (linearly) decreasing distribution in terms of the frequency of the distinct process instances, while for the uniform log each distinct process instance accounts for a frequency of 10.

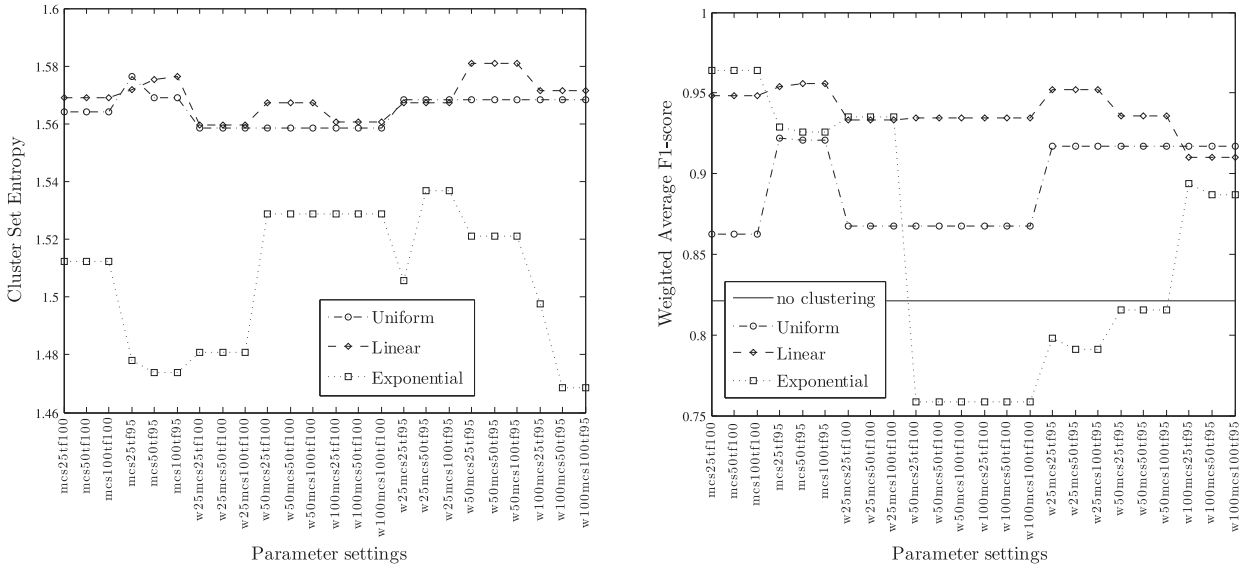
5.1.2 Results

Within this controlled environment, different parameter settings for the ActiTraC algorithm are evaluated by verifying how well ActiTraC is able to identify the different classes of behavior as well as by investigating the accuracy improvements realized by clustering traces. Moreover, the performance of ActiTraC in this environment is compared to seven existing trace clustering techniques.

Parameter settings. For ActiTraC, three parameters are considered essential: *window size*, *minimum cluster size*, and *target fitness*. As such, different values for these parameters were evaluated according to cluster set entropy and

1. http://www.promtools.org/prom6/.

2. http://www.cpntools.org.



(a) H_{CS} (lower is better) (b) $\mathrm{F1}_{W.A.}$ (higher is better)

Fig. 3. Cluster set entropy and weighted average F1-scores for different parameter settings of ActiTraC and for three distinct event logs with varying frequency distributions of the dpi's.

weighted average F1-scores for the three artificial event logs, as shown in Fig. 3. When taking into account the ability of ActiTraC to distinguish between the induced clusters, the performance is rather weak. Especially for the uniform and linear event log, the cluster set entropy is very close to the maximal entropy of 1.585. On the other hand, the exponential case provides us with some more insights into the behavior of the algorithm. For instance, for the pure frequency-based selective sampling (the first six settings on the x-axis), a decrease in terms of the target fitness has a positive effect on the entropy. For the MRA-based sampling procedure, the effects of window size and minimum cluster size depend on the target fitness. In case of a target fitness of 1, a decreased window size tends to improve cluster set entropy, while for a decreased target fitness, an increase of the window size tends to affect cluster set entropy positively.

Next to entropy, the accuracy results realized with different values of the ActiTraC parameters are detailed in Fig. 3b. For the more challenging uniform and linear logs, it holds that a decrease in target fitness has a positive effect on the F1-scores. This is mostly due to the difficulty for ActiTraC to join high numbers of traces in one single process model with perfect fitness. This is because the clusters formed by ActiTraC with the target fitness set to 1 are founded on a limited set of traces which causes the underlying models to be inaccurate for the entire set of traces in the cluster. When decreasing the target fitness, clusters are created based on a higher number of traces, which has a positive effect on the F1-score. This shows that event logs consisting of more challenging control-flow behavior for process discovery techniques might benefit from decreasing the target fitness. Note that the solid, nonmarked, horizontal line denotes the average F1-score over the three event logs for the baseline scenario where no clustering is performed.

In the case of a more or less exponential distribution of trace frequencies, the effects of decreasing the target fitness are reversed. For both selective sampling strategies, the best results can be obtained by setting the target fitness to 1. This is coherent with the design of ActiTraC which employs trace frequencies as a steering information source for selecting traces and optimizing the underlying process models. Finally, note that the effects of the minimum cluster size are limited because of the difficulty of the artificially generated event logs. Due to the pure randomness of trace generation, the ActiTraC algorithm is unable to find a sufficient amount of traces that can be clustered with the selected target fitness. In other situations, enlarging the minimum cluster size could be useful to increase the total amount of traces on which the underlying process models are learned from.

Benchmark performance. Next to the evaluation of different parameter settings for ActiTraC, the closed environment is employed to benchmark the performance with other clustering techniques. Based on the previous analysis, standard parameter settings for both $\mathrm{ActiTraC}^{\mathrm{freq}}$ and $\mathrm{ActiTraC}^{\mathrm{MRA}}$ are defined as follows: target fitness is set to 1, minimum cluster size is set to 25 percent and for $\mathrm{ActiTraC}^{\mathrm{MRA}}$ a window size of 25 percent is set. Despite the fact that in some situations, it might be useful to change these values, standard parameters are used in the remaining analyses. Note that these settings are not *optimal* for each event log individually. The development of a parameter optimization strategy that can deal with possibly large computation times required for clustering and evaluation metric calculation is out of the scope of this study.

ActiTraC is compared to seven clustering techniques: MR and MRA [13], GED and LED [7], bag of activities (BOA) and 3-gram [12], and the Markov chain clustering technique from [15]. It should be noted that agglomerative hierarchical clustering with minimum variance is used as the underlying clustering technique for BOA, 3-gram, GED, LED, MR, and MRA. Despite the fact that other underlying clustering techniques might increase performance, this is

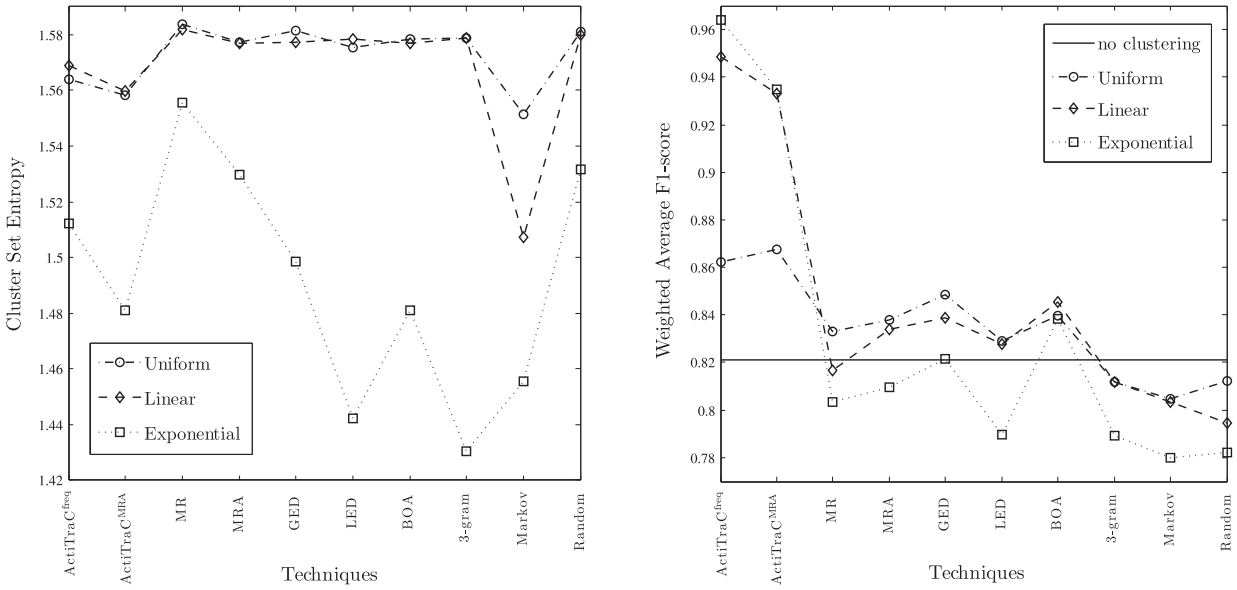


Fig. 4. Comparison of ActiTraC with other trace clustering techniques according to cluster set entropy and weighted average F1-scores.

not considered in this study. For BOA, 3-gram, MR, and MRA, the best performing parameter settings are employed for each evaluation criterion and for each event log individually by varying the vector representation (i.e., binary or numeric) and the underlying distance metric (i.e., F-score similarity or euclidean distance). The application of feature filtering is not considered. Finally, the average results of a fivefold repetition of randomly clustering traces into different groups is also added (Random).

The results of the comparative analysis are depicted in Fig. 4. Taking into account the cluster set entropy, it is concluded that the benchmark techniques tend to face the same difficulties as ActiTraC with respect to distinguishing the different classes of behavior. Only the Markov clustering technique outperforms the ActiTraC algorithms on every artificial data set. However, with a best cluster set entropy of only 1.44 (3-gram), the main conclusion of this experiment is that all assessed clustering techniques are incapable of adequately detecting the predetermined classes. It is pointed out that clustering techniques might divide traces into different clusters based on other domain-specific criteria. Accordingly, we cannot draw general conclusions with respect to the domain relevancy of different clustering techniques.

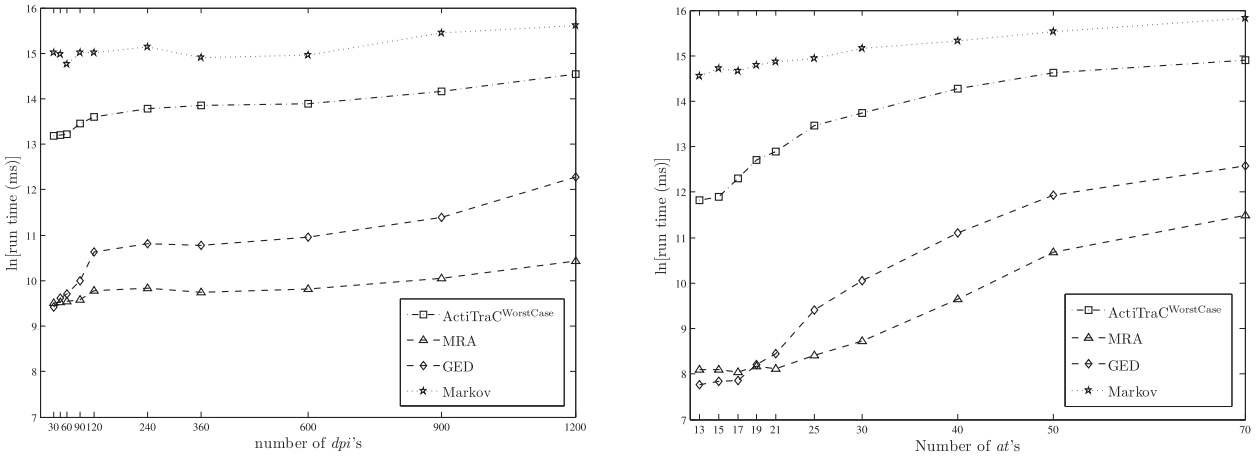
Fig. 4b presents the accuracy results in terms of the weighted average F1-score. Note that this metric is calculated by first applying HeuristicsMiner (with standard parameter settings) to the clusters and then translating the resulting model into a Petri net by making use of a translation plugin in ProM. Next, recall and precision metrics are calculated and combined into an individual F1-score for each cluster. As detailed in Section 4.2.1, these F1-scores can be weighted so as to compute the weighted average F1-score for each of the solutions of the clustering techniques. Note that for ActiTraC, the process models underlying each of the clusters are calculated within the clustering algorithm.

In the figure, the effects of bridging the gap between clustering bias and evaluation bias are clearly observable. Even for the most challenging setting for ActiTraC, i.e., the uniform event log where there is no frequency difference between the traces and thus the selection is essentially random for $\mathrm{ActiTraC}^{\mathrm{freq}}$ and purely MRA-driven for $\mathrm{ActiTraC}^{\mathrm{MRA}}$, both techniques outperform every other available clustering technique, even with suboptimal parameter settings. As soon as trace frequencies start to differ among traces, this effect is magnified.

5.2 Scalability

Before further assessing the capabilities of ActiTraC on real-life event logs, its scalability is examined. Note that in Section 3.5, it was already pointed out that some essential design choices of the algorithm, namely its underlying process discovery technique and its underlying fitness score are mainly chosen for scalability reasons. Regarding complexity, the core of the ActiTraC algorithm scales linearly in terms of the number of *dpi*'s as well as in terms of the number of clusters. However, the overall computational complexity of our algorithm depends largely on the underlying techniques, i.e., the process discovery technique, the fitness calculation and the trace selection procedure. As such, considering the current choices, ActiTraC scales linearly in terms of the number of clusters and quadratically in terms of *dpi*'s and in terms of activity types (*at*).

Fig. 5 details a scalability analysis of ActiTraC for its worst case scenario. This scenario is determined by setting the window size to 1 and the target fitness to $-\infty$. In this case, none of the implemented greedy heuristics are used so that a maximal amount of selective sampling procedures needs to be executed. The scalability assessment consists of the creation of 100 artificial event logs with varying number of activity types and varying number of distinct process instances. Note that with these settings, the effect of the underlying distribution of trace frequencies is



(a) Distinct Process Instances (b) Activity Types

Fig. 5. Scalability comparison of ActiTraC for its worst case scenario by varying the number of distinct process instances and the number of activity types.

canceled out. By focusing on distinct process instances and activity types, the two main complexity criteria for ActiTraC are taken into account.

Fig. 5a depicts the runtime results of four clustering algorithms varying the number of distinct process instances. Note that the plot uses a logarithmic scale. Each marker in this figure represents the average calculated over 10 different event logs with varying number of activity types. Fig. 5b mirrors this setting for the number of activity types averaging out over 10 logs with varying number of distinct process instances. The results clearly show an increased complexity for ActiTraC as compared to contemporary clustering algorithms. Note that MRA is shown as a representative for both GED as well as for all other vector space approaches because of very similar performance results. In conclusion, ActiTraC in its worst case scenario requires up to 50 times as much computational resources in comparison to vector space approaches. Nonetheless, ActiTraC does outperform the Markovian model-based clustering technique, which is the most similar of all other clustering techniques in terms of the clustering procedure. Note that the scalability assessment was performed on a standard stand-alone PC with a 2.83-Ghz quad-core CPU and 3-GB of RAM.

5.3 Real-Life Event Logs

The final part of the experimental evaluation of ActiTraC consists of its application to four real-life event logs. Table 1 shows some basic statistics and a description of the event logs used. It is important to note that the event logs originate from flexible environments exhibiting a large variety of process behavior, as illustrated with the metrics in the table. Because in real-life settings, ex ante classes of behavior are unknown, the use of entropy-based evaluation is infeasible. Therefore, the evaluation is centered on two dimensions, i.e., accuracy in terms of the weighted average F1-score and complexity in terms of the average place/transition connection degree.

The experiments are set up in a similar way as the comparative assessment in the controlled environment. For the four real-life event logs, 10 different trace clustering approaches were applied varying the maximal amount of clusters from 3 to 5. For both $\mathrm{ActiTraC}^{\mathrm{freq}}$ and $\mathrm{ActiTraC}^{\mathrm{MRA}}$, the standard parameters are used while for the other techniques the best performing parameter settings are employed for each data set and each cluster size individually. This setup emphasizes ActiTraC's robustness.

5.3.1 Accuracy Results

The accuracy results are presented in Fig. 6. The solid, nonmarked, horizontal line denotes the base case with no trace clustering where HeuristicsMiner is applied to the whole event log. In general, it is concluded that the ActiTraC algorithms significantly outperform the reference clustering algorithms with respect to process model accuracy. We observe consistent performance improvements considering both the different event logs and the varying number of clusters.

TABLE 1
Description of the Real-Life Event Logs with Following Characteristics: The Number of Process Instances (# pi), Number of Events (# ev), the Number of Activity Types (# at), and the Number of Distinct Process Instances (# dpi)

Label	Event log properties				Organization	Process description
	# pi	# ev	# at	# dpi		
KIM	24.770	124.217	18	1.174	KU Leuven	Helpdesk process of the ICT service
MCRM	956	11.218	22	212	Manufacturing company	CRM process
TSL	17.812	83.286	42	1.908	Telecom company	Second-line CRM process
ICP	12.391	65.653	70	1.411	Insurance company	Incoming document handling

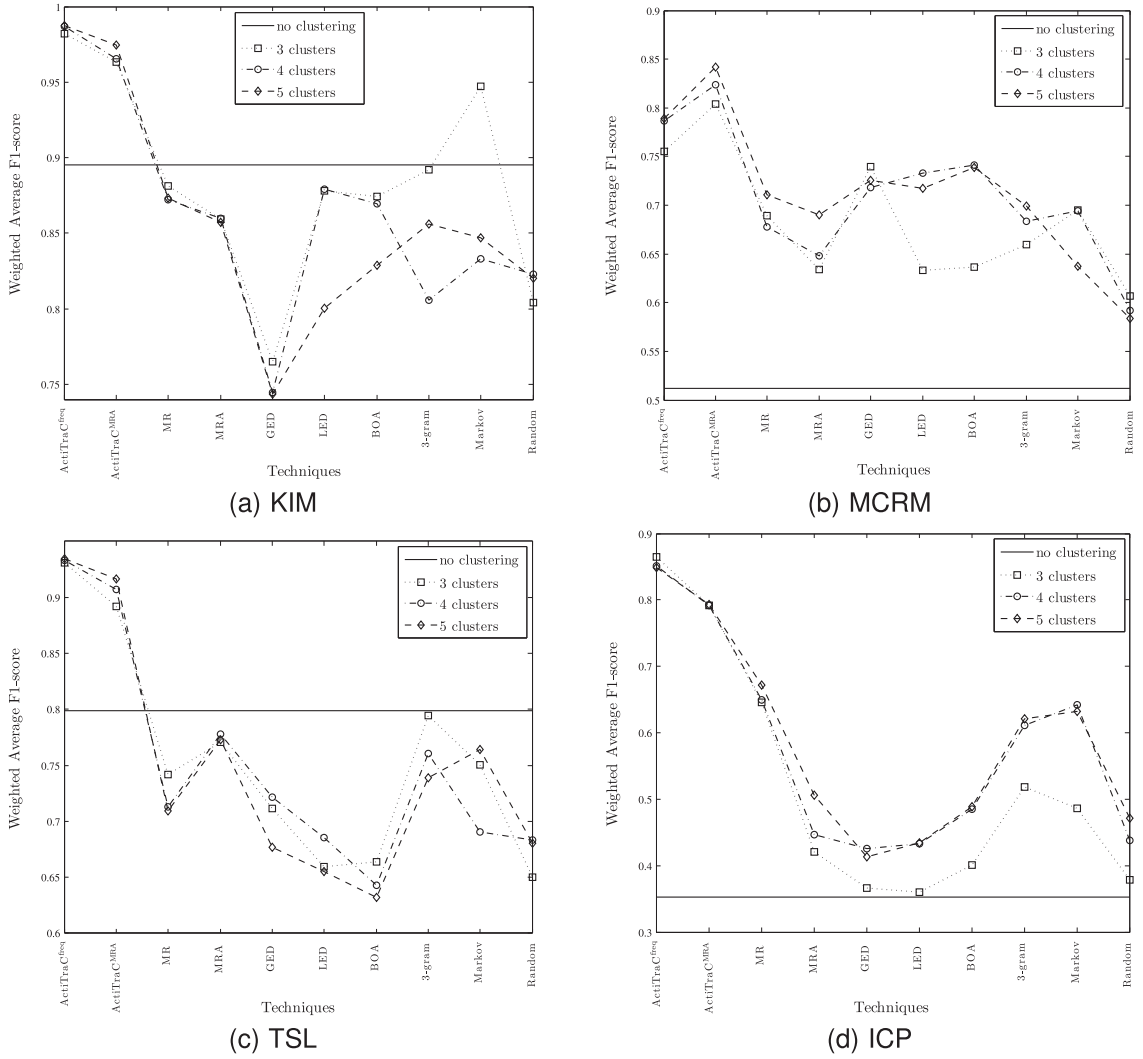


Fig. 6. Weighted average F1-scores for the four real-life event logs. Note that the ActiTraC algorithms clearly outperform the other clustering techniques in all the benchmarks (higher is better).

Remark also that for many of the existing trace clustering techniques, the weighted average F1-score is rather poor. As such, for event logs KIM and TSL, almost all other techniques perform worse than the case of not clustering at all. Also, in many situations, these techniques cannot significantly outperform the random clustering of traces. This is clear evidence for the initial observation that there is a strong divergence between the clustering bias and the evaluation bias for these techniques. As for ActiTraC, it is concluded that in most cases, the pure frequency-based selective sampling strategy will slightly outperform the MRA-based selective sampling strategy from an accuracy perspective. However, the smaller MCRM log shows that it can be advantageous to force the selection of more similar traces first.

5.3.2 Complexity Results

Our evaluation approach also consists of an assessment of the comprehensibility of the discovered process models. As explained in Section 4, the average place/transition connection degree is employed as a quantification of process model complexity. The results of this complexity analysis are presented in Fig. 7. For these plots, the lower the value, the better the performance because a decrease in average place/transition connection degree signifies a less complex process model.

From the figure, it can be seen that also from a complexity viewpoint, the results clearly favor the ActiTraC algorithms. For the four real-life logs, the additional decrease in complexity of the Petri net models as compared to the other clustering techniques is significant. With respect to the difference between the ActiTraC techniques themselves, there is a slight indication that MRA-based sampling has the potential to create clusters for which the underlying process models are less complex. Only for the ICP event log, this pattern cannot be observed.

5.3.3 Scalability

Because the computational complexity of ActiTraC is a crucial factor determining its applicability in practice, Table 2 provides an overview of the runtimes of the different techniques for the creation of four clusters for each of the real-life event logs. From these results, it is concluded that the greedy heuristics underlying the ActiTraC algorithm are effective in terms of restricting the computational requirements.

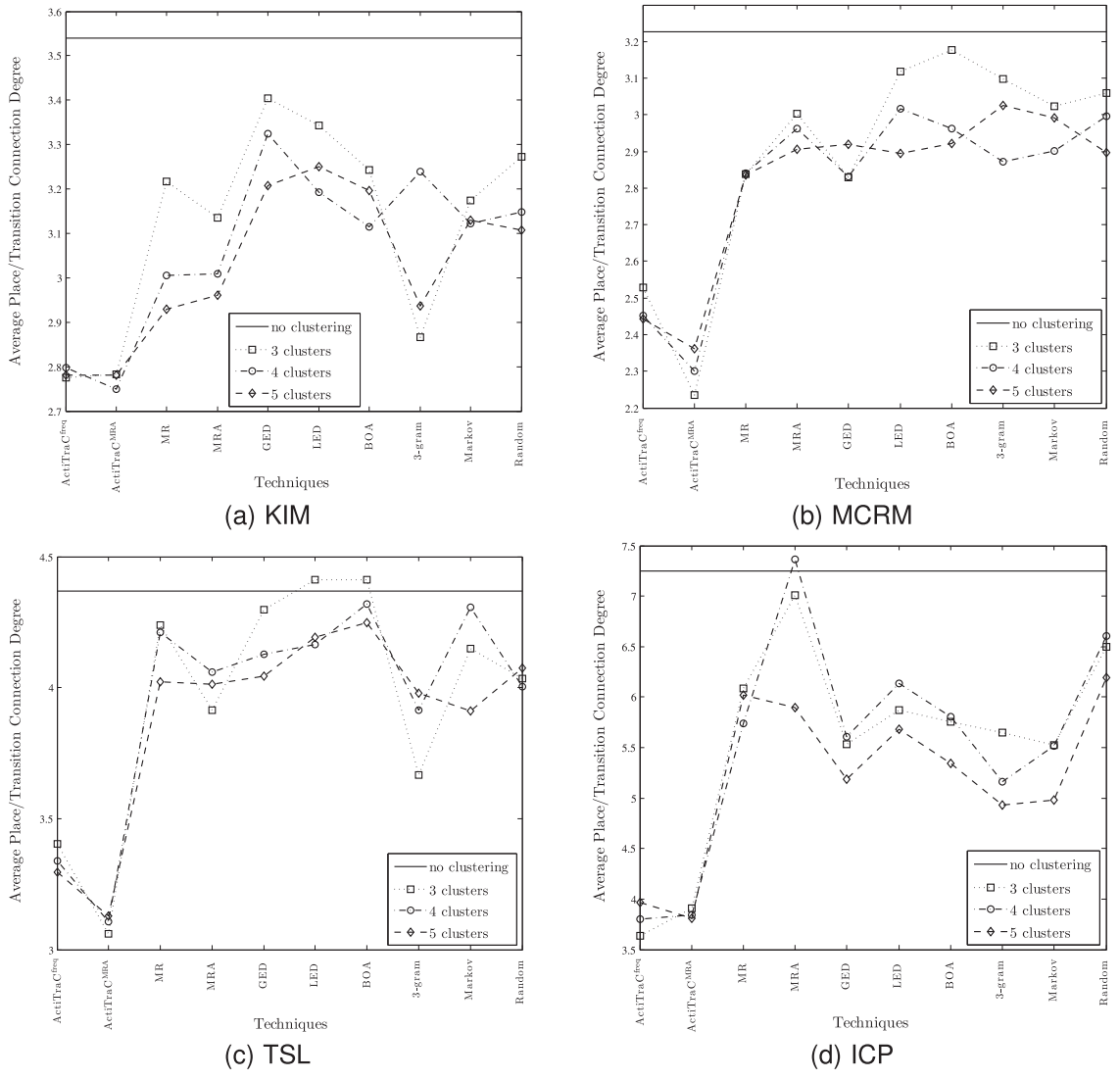


Fig. 7. Average place/transition connection degree for the four real-life event logs. Note that the ActiTraC algorithms clearly outperform the other clustering techniques in all the benchmarks (lower is better).

TABLE 2
Average Runtimes with 95 Percent Confidence Interval for the Real-Life Event Logs with Four Clusters

Technique	KIM	MCRM	TSL	ICP
ActiTraC^{freq}	00:04:39,912 ± 6,568s	00:01:01,739 ± 2,163s	00:05:02,383 ± 15,694s	00:05:48,682 ± 12,207s
ActiTraC^{MRA}	00:02:58,496 ± 1,306s	00:00:13,328 ± 0,052s	00:06:11,680 ± 1,671s	00:10:18,691 ± 5,099s
ActiTraC^{WorstCase}	01:27:58,290 ± 9,603s	00:01:18,764 ± 0,041s	02:04:35,266 ± 48,694s	01:02:36,146 ± 8,389s
MR	00:01:16,214 ± 0,525s	00:00:01,907 ± 0,005s	00:03:59,391 ± 1,070s	00:01:12,726 ± 0,337s
MRA	00:00:33,384 ± 0,376s	00:00:01,778 ± 0,014s	00:01:59,582 ± 0,501s	00:00:56,089 ± 0,328s
GED	00:00:41,952 ± 0,068s	00:00:03,578 ± 0,046s	00:01:10,673 ± 0,467s	00:00:31,556 ± 0,114s
LED	00:00:07,600 ± 0,058s	00:00:00,391 ± 0,001s	00:00:22,171 ± 1,043s	00:00:09,297 ± 0,087s
BOA	00:00:06,472 ± 0,223s	00:00:00,244 ± 0,010s	00:00:22,324 ± 0,184s	00:00:11,309 ± 0,270s
3-gram	00:00:21,862 ± 0,070s	00:00:00,515 ± 0,000s	00:02:54,665 ± 0,847s	00:01:33,392 ± 0,384s
Markov	08:22:40,000 ± 9334,281s	00:00:51,000 ± 13,917s	04:35:14,000 ± 4042,004s	01:42:56,000 ± 756,172s

6 Discussion

To conclude, the presented trace clustering technique achieves its goal to improve currently available trace clustering techniques when evaluation is considered from a process discovery perspective. Our approach changes the objective of traditional trace clustering to find an optimal distribution in terms of grouping similar traces together to the goal of solving the problem of finding an optimal distribution of traces so as to maximize the combined accuracy of the underlying process models. We think that in many situations, the latter objective is useful because the challenge of finding a criterion that separates process instances according to their intrinsic similarity is often made difficult due to the data explosion problem. This problem was not only demonstrated with the experiment in the supervised environment but is also recognized in [32] where outlier detection is proposed as a solution to the problem.

Nevertheless, it is acknowledged that grouping homogeneous traces together is useful. Therefore, ActiTraC has been provided with the flexibility to take into account any

kind of distance function between instances. With respect to addressing the challenge of combined accuracy and homogeneity optimization, we are convinced that a more comprehensive benchmarking environment for trace clustering techniques should be devised. Currently, such a framework is not available in the process mining literature.

A final element of discussion is cluster characterization. For trace clustering to be a genuine approach toward solving the complexity problem arising from highly unstructured event logs, process analysts should be able to delineate the discovered clusters to give an interpretation to the solution. In future research, we intend to investigate how for instance rule learning algorithms can be employed to ex post characterize the differences between clusters of process instances. This might provide us with some more insights toward defining a measure of inter-cluster dissimilarity, a feature that is currently lacking from a cluster evaluation perspective.

7 Conclusion

In this paper, we have proposed a new approach to trace clustering. Trace clustering is one possible strategy to resolve the problem that currently available process discovery algorithms are unable to discover accurate and comprehensible process models out of event logs stemming from highly flexible environments. The technique, called ActiTraC, has its foundations in the observation that currently available trace clustering algorithms suffer from a severe divergence between the clustering bias and the evaluation bias. This problem is resolved with an active learning inspired approach that centers on optimizing the combined process model accuracy. In the experimental evaluation, both artificial as well as real-life event logs were used to show that ActiTraC, even without optimization of its parameters, significantly improves the accuracy and complexity of the process models underlying the discovered trace clusters as compared to existing trace clustering techniques.

Acknowledgments

The authors would like to thank the Flemish Research Council for financial support under Odysseus grant B.0915.09 and KU Leuven for OT/10/010. Also a special thanks to Niels Lambrigts for his support with the development of the ActiTraC plugin.

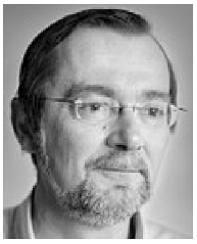
References

[1] W.M.P. van der Aalst, *Process Mining - Discovery, Conformance and Enhancement of Business Processes.* Springer, 2011.
[2] C.W. Günther, "Process Mining in Flexible Environments," PhD dissertation, TU Eindhoven, 2009.
[3] M. Song and W.M.P. van der Aalst, "Towards Comprehensive Support for Organizational Mining," *Decision Support Systems,* vol. 46, no. 1, pp. 300-317, 2008.
[4] W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster, "Workflow Mining: Discovering Process Models from Event Logs," *IEEE Trans. Knowledge Data Eng.,* vol. 16, no. 9, pp. 1128-1142, Sept. 2004.
[5] M. Sole and J. Carmona, "Region-Based Foldings in Process Discovery," *IEEE Trans. Knowledge Data Eng.,* vol. 25, no. 1, pp. 192-205, Jan. 2013.
[6] S. Goedertier, J. De Weerdt, D. Martens, J. Vanthienen, and B. Baesens, "Process Discovery in Event Logs: An Application in the Telecom Industry," *Applied Soft Computing,* vol. 11, no. 2, pp. 1697-1710, 2011.
[7] R.P. Jagadeesh Chandra Bose and W.M.P. van der Aalst, "Context Aware Trace Clustering: Towards Improving Process Mining Results," *Proc. SIAM Int'l Conf. Data Mining (SDM),* pp. 401-412, 2009.
[8] G.M. Veiga and D.R. Ferreira, "Understanding Spaghetti Models with Sequence Clustering for Prom," *Proc. Int'l Business Process Management Workshops,* pp. 92-103, 2010.
[9] R.P. Jagadeesh Chandra Bose and W.M.P. van der Aalst, "Abstractions in Process Mining: A Taxonomy of Patterns," *Proc. Seventh Int'l Conf. Business Process Management (BPM),* pp. 159-175, 2009.
[10] C.W. Günther and W.M.P. van der Aalst, "Fuzzy Mining - Adaptive Process Simplification Based on Multi-Perspective Metrics," *Proc. Fifth Int'l Conf. Business Process Management (BPM),* pp. 328-343, 2007.
[11] G. Greco, A. Guzzo, L. Pontieri, and D. Saccà, "Discovering Expressive Process Models by Clustering Log Traces," *IEEE Trans. Knowledge Data Eng.,* vol. 18, no. 8, pp. 1010-1027, Aug. 2006.
[12] M. Song, C.W. Günther, and W.M.P. van der Aalst, "Trace Clustering in Process Mining," *Proc. Int'l Business Process Management Workshops,* pp. 109-120, 2008.
[13] R.P. Jagadeesh ChandraBose and W.M.P. van der Aalst, "Trace Clustering Based on Conserved Patterns: Towards Achieving Better Process Models," *Proc. Int'l Business Process Management Workshops,* pp. 170-181, 2009.
[14] V.I. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions and Reversals," *Soviet Physics Doklady,* vol. 10, pp. 707-710, 1966.
[15] D.R. Ferreira, M. Zacarias, M. Malheiros, and P. Ferreira, "Approaching Process Mining with Sequence Clustering: Experiments and Findings," *Proc. Fifth Int'l Conf. Business Process Management (BPM),* pp. 360-374, 2007.
[16] I. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White, "Model-Based Clustering and Visualization of Navigation Patterns on a Web Site," *Data Mining and Knowledge Discovery,* vol. 7, no. 4, pp. 399-424, 2003.
[17] B. Settles, "Active Learning Literature Survey," Computer Sciences Technical Report 1648, Univ. of Wisconsin-Madison, 2009.
[18] D. Angluin, "Queries and Concept Learning," *Machine Learning,* vol. 2, no. 4, pp. 319-342, 1987.
[19] D. Cohn, L. Atlas, and R. Ladner, "Improving Generalization with Active Learning," *Machine Learning,* vol. 15, pp. 201-221, 1994.
[20] D. Martens, B. Baesens, and T.V. Gestel, "Decompositional Rule Extraction from Support Vector Machines by Active Learning," *IEEE Trans. Knowledge Data Eng.,* vol. 21, no. 2, pp. 178-191, Feb. 2009.
[21] T. Hofmann and J.M. Buhmann, "Active Data Clustering," *Proc. Neural Information Processing Systems Conf. (NIPS),* 1997.
[22] A.J.M.M. Weijters, W.M.P. van der Aalst, and A.K. Alves de Medeiros, "Process Mining with the Heuristicsminer Algorithm," TU Eindhoven, BETA Working Paper Series 166, 2006.
[23] J. De Weerdt, M. De Backer, J. Vanthienen, and B. Baesens, "A Multi-Dimensional Quality Assessment of State-of-the-Art Process Discovery Algorithms Using Real-Life Event Logs," *Information Systems,* vol. 37, no. 7, pp. 654-676, 2012.
[24] A.J.M.M. Weijters and W.M.P. van der Aalst, "Rediscovering Workflow Models from Event-Based Data Using Little Thumb," *Integrated Computer-Aided Eng.,* vol. 10, no. 2, pp. 151-162, 2003.
[25] A. Rozinat and W.M.P. van der Aalst, "Conformance Checking of Processes Based on Monitoring Real Behavior," *Information Systems,* vol. 33, no. 1, pp. 64-95, 2008.
[26] J. De Weerdt, M. De Backer, J. Vanthienen, and B. Baesens, "A Critical Evaluation Study of Model-Log Metrics in Process Discovery," *Proc. Int'l Business Process Management Workshops,* pp. 158-169, 2010.
[27] J. De Weerdt, M. De Backer, J. Vanthienen, and B. Baesens, "A Robust F-Measure for Evaluating Discovered Process Models," *Proc. IEEE Symp. Computational Intelligence and Data Mining (CIDM),* pp. 148-155, 2011.
[28] S. Goedertier, D. Martens, J. Vanthienen, and B. Baesens, "Robust Process Discovery with Artificial Negative Events," *J. Machine Learning Research,* vol. 10, pp. 1305-1340, 2009.

[29] K.B. Lassen and W.M.P. van der Aalst, "Complexity Metrics for Workflow Nets," *Information and Software Technology*, vol. 51, no. 3, pp. 610-626, 2009.
[30] J. Mendling, H.A. Reijers, and W.M.P. van der Aalst, "Seven Process Modeling Guidelines (7 pmg)," *Information and Software Technology*, vol. 52, no. 2, pp. 127-136, 2010.
[31] T. Murata, "Petri Nets: Properties, Analysis and Applications," *Proc. IEEE*, vol. 77, no. 4, pp. 541-580, Apr. 1989.
[32] F. Folino, G. Greco, A. Guzzo, and L. Pontieri, "Mining Usage Scenarios in Business Processes: Outlier-Aware Discovery and Run-Time Prediction," *Data Knowledge Eng.*, vol. 70, no. 12, pp. 1005-1029, 2011.

Jochen De Weerdt received the MSc degree in business economics—information systems engineering from KU Leuven, Belgium. He is currently working as a scientific researcher in the Department of Decision Sciences and Information Management at KU Leuven. His research interests include data mining, process mining, and web intelligence.

Seppe vanden Broucke received the MSc degree in business economics—information systems engineering from the Department of Decision Sciences and Information Management at KU Leuven. He is working toward the PhD degree at KU Leuven in such areas as business process mining and management, data mining, and event sequence analysis.



Jan Vanthienen received the PhD degree in applied economics from the KU Leuven, Belgium. He is a full professor of information systems with the Department of Decision Sciences and Information Management, KU Leuven. He is the author or coauthor of numerous papers published in international journals and conference proceedings. His current research interests include information and knowledge management, business intelligence and business rules, and information systems analysis and design.

Bart Baesens is an associate professor at KU Leuven, Belgium, and a lecturer at the University of Southampton, United Kingdom. He has done extensive research on predictive analytics, data mining, customer relationship management, Web analytics, fraud detection, and credit risk management. His findings have been published in well-known international journals (e.g., *Machine Learning*, *Management Science*, *IEEE Transactions on Neural Networks*, *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Evolutionary Computation*, *Journal of Machine Learning Research*, etc.) and presented at top international conferences.