Compound Trace Clustering to Generate Accurate and Simple Sub-Process Models

Yaguang Sun[1(✉)], Bernhard Bauer[1], and Matthias Weidlich[2]

[1] Software Methodologies for Distributed Systems, University of Augsburg, Augsburg, Germany
{yaguang.sun,bernhard.bauer}@informatik.uni-augsburg.de
[2] Humboldt-Universität zu Berlin, Berlin, Germany
matthias.weidlich@hu-berlin.de

Abstract. Business process model discovery targets the construction of conceptual models from event data that has been recorded during the execution of a business process. While a plethora of discovery techniques have been proposed in the literature, most existing techniques fail to cope with complex control-flow patterns as they are observed in event logs of highly flexible processes. In this paper, we follow the idea of splitting-up an event log into sub-logs, before applying process model discovery. This yields a set of sub-process models, one per sub-log, each describing a major variant of the business process. Unlike existing techniques, our clustering approach is guided by the result of model discovery: It first optimises the average complexity of the resulting models, before improving the accuracy of each model in isolation. Our experimental evaluation highlights that our approach yields more accurate sub-process models (that are of comparatively low complexity) than state-of-the-art trace clustering techniques.

Keywords: Business process mining · Process model discovery · Trace clustering · Model fitness improvement · Model complexity reduction

1 Introduction

Manual elicitation of business process models is regarded a complex, time consuming, and error-prone task. In recent years, therefore, techniques for automated business process model discovery (BPMD) have been developed, which aim at the construction of conceptual models from event data that has been recorded during the execution of a business process [1]. The starting point for BPMD is an event log that is generated by information systems and contains information on *traces*. A trace is a sequence of events that denote activity executions for a particular instance of a business process.

While a large number of BPMD techniques have been described in the literature, see [7,11,18], most existing approaches fail to cope with complex control-flow patterns in real-life event logs, which usually stem from business processes

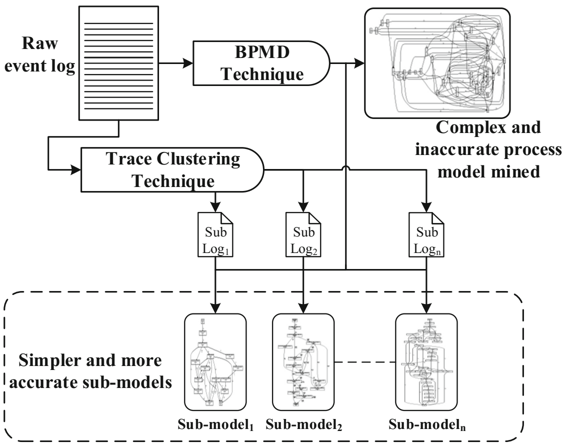

Raw event log
BPMD Technique
Complex and inaccurate process model mined
Trace Clustering Technique
Sub Log1
Sub Log2
Sub Logn
Simpler and more accurate sub-models
Sub-model1
Sub-model2
Sub-modeln

Fig. 1. The basic setting of using trace clustering in business process model discovery.

implemented in highly flexible environments, e.g., healthcare, customer relationship management (CRM), and product development [6]. For such processes, the phenomenon of *'spaghetti-like'* process models has been described in multiple case studies. Such models are often inaccurate and too complex to be interpreted by domain experts [4], and thus of limited use. This problem is largely due to the presence of diverse variants of a business process within a single event log [3].

Against this background, it has been argued that *trace clustering* [2–6,8–10] shall be applied before BPMD. As outlined in Fig. 1, an event log is first split into sub-logs, each containing traces of similar structure. Afterwards, BPMD techniques are applied to each of the generated sub-logs to obtain a set of sub-process models that provide a more accurate and comprehensible view on the business process. However, existing trace clustering techniques are largely decoupled from process model discovery. They are primarily guided by the similarity of traces in an event log, but are agnostic to the impact of clustering decisions on the quality of the discovered models. Consequently, applying traditional trace clustering in BPMD may yield inaccurate sub-process models.

In this paper, we therefore put forward a new trace clustering technique named *Compound Trace Clustering* (CTC). It considers the accuracy and complexity of the resulting sub-process models during the clustering procedure. More specifically, it first optimises the average complexity of the sub-process models, before the accuracy of each model is improved separately.

In the remainder of this paper, we first exemplify the issues of applying traditional trace clustering in BPMD with experimental results (Sect. 2) and introduce basic formal notions and notations (Sect. 3). We then elaborate on the details of the proposed CTC technique (Sect. 4). To test the efficiency of our method, we carried out a comprehensive evaluation with four real-world event logs (Sect. 5). As part of that, we also compared CTC with six traditional trace clustering techniques. Finally, we review related work (Sect. 6) and conclude (Sect. 7).

2 Issues of Traditional Trace Clustering in BPMD

Existing trace clustering techniques are decoupled from business process model discovery and focus on the detection of similarity between traces in a given event log. As such, they largely neglect the implications of certain clustering decisions on the accuracy of the sub-process models derived per sub-log [6]. Some of the resulting sub-process models are therefore likely to be of low quality.

We illustrate this issue with experimental insights obtained for the event log of the loan and overdraft approval process [12] that has been published as part of the Business Process Intelligence Challenge (BPIC) in 2012. Using two traditional trace clustering techniques, namely GED [4] and sequence clustering (SCT) [5], and setting the number of generated sub-logs to five, yields the results shown in Table 1. For each method, the table lists the number of traces in the respective sub-logs as well as the quality of the sub-process model discovered from it. Models have been constructed with the Flexible Heuristics Miner (FHM) [11] and accuracy is measured in terms of fitness [13], i.e., the amount of behaviour present in the log that is covered by the discovered model.

Table 1. The information about the sub-process models mined from the sub-logs of LOA generated by two traditional trace clustering techniques.

Method	Metrics	Model of sub-log 1	Model of sub-log 2	Model of sub-log 3	Model of sub-log 4	Model of sub-log 5
GED	Fitness	0.9718	0.9959	0.8049	0.5193	0.6197
	#Traces	1509	1607	8073	784	1114
SCT	Fitness	0.9095	0.8436	0.9636	0.932	0.7828
	#Traces	2091	1839	1740	2765	4652

The results illustrate that both trace clustering techniques will generate one or more sub-logs, for which the discovered sub-process models have low fitness. For example, the fitness of the model discovered from sub-log 4 as constructed by GED is only 0.5193, meaning that a large part of the behaviour of the sub-log cannot be replayed in the model. For the case of SCT, we observe that the model generated for sub-log 5 has a comparatively low fitness value of 0.7828.

The above results exemplify that conducting trace clustering independent of business process model discovery may yield sub-process models of low quality. In the remainder, we will therefore present a new clustering mechanism that helps to generate accurate and simple sub-process models.

3 Preliminaries

In this section, we introduce fundamental concepts and notations needed to define our approach to compound trace clustering.

Let I be a set of items (we will later consider activities as items), $\mathcal{S}(I)$ be the set of all finite sequences over I. A sequence $s \in \mathcal{S}(I)$ of length m is denoted $\langle it_1, it_2, \ldots, it_m \rangle$, where each element it_k is an item from I. For two sequences $X = \langle x_1, x_2, \ldots, x_l \rangle$ and $Y = \langle y_1, y_2, \ldots, y_q \rangle$ from $\mathcal{S}(I)$, of length l and q, respectively, X is a sub-sequence of Y, denoted as $X \sqsubseteq Y$, if $1 \leq p_1 < p_2 < \cdots < p_l \leq q$ such that $x_1 = y_{p_1}, x_2 = y_{p_2}, \ldots, x_l = y_{p_l}$.

We also need notions related to frequent sequences. Let DS be a set (or database) of sequences. By $support(seq)$, we denote the number of sequences in DS that contain the sequence seq as a sub-sequence. Given a minimum support value min_sup, with $0 < min\_sup < 1$, a sequence seq is called a *sequential pattern* (or a frequent sequence), if $support(seq) \geq min\_sup \times |DS|$. The set of sequential patterns, SP, consists of all sub-sequences of DS, for which the support values are no less than $min\_sup \times |DS|$. Acknowledging that SP contains partly redundant information in terms of sequential patterns that are contained in other patterns, we also define the set of *closed sequential patterns* as $CSP = \{\alpha \in SP \mid \nexists\, \beta \in SP : \alpha \sqsubseteq \beta \wedge support(\alpha) = support(\beta)\}$. Many algorithms for the detection of sequential patterns have been proposed in the literature, see [15, 16]. For our purposes, it suffices to abstract from a specific algorithm for closed sequential pattern mining, which we assume to be given as $\Gamma : DS^+ \xrightarrow{min\_sup} CSP^+$, where DS^+ is the universe of sequence databases, CSP^+ is the universe of sets of closed sequential patterns, and min_sup is a minimum support value.

Next, we turn to the notion of an event log, as recorded by information systems during the execution of a business process. Let A be the universe of activities of a business process. Then, an event e denotes the execution of an instance of a particular activity $a \in A$. With E as the universe of such events, we define an event log as follows.

Definition 1 *(Trace, Event Log). A* trace $t \in \mathcal{S}(E)$ *is a sequence of events. An* event log L *is a non-empty multiset of traces.*

For instance, $L = \{\langle a, b, c, d \rangle^{23}, \langle a, c, b, d \rangle^{16}\}$ denotes an event log built of 156 events that refer to four activities (a, b, c and d). The events are part of 39 traces, with the variant $\langle a, b, c, d \rangle$ appearing 23 times, while the variant $\langle a, c, b, d \rangle$ appears 16 times in L.

With L^+ as the universe of event logs and M^+ as the universe of process models, $\Lambda : L^+ \rightarrow M^+$ is a BPMD algorithm. To evaluate the result quality of BPMD, we further consider a process model complexity measure, $\Sigma : M^+ \rightarrow \mathbb{R}$.

4 Compound Trace Clustering for Process Discovery

This section presents a novel trace clustering technique named *Compound Trace Clustering* (CTC) for process discovery. An overview of our approach is given in

Fig. 2. In essence, we proceed in two stages. In the first stage, the given event log is split into sub-logs, so that the sub-process models derived from these logs with some business process model discovery technique have an optimal average complexity. In a second stage, the accuracy of these sub-models created in stage 1 is assessed and, if needed, improved by employing an algorithm proposed in our earlier work [17]. Below, we first present details of our novel trace clustering technique for stage 1 (Sect. 4.1), before providing a short summary of the algorithm for improving model accuracy in stage 2 (Sect. 4.2). Finally, we integrate these building blocks and define the complete algorithm for compound trace clustering for process discovery (Sect. 4.3).

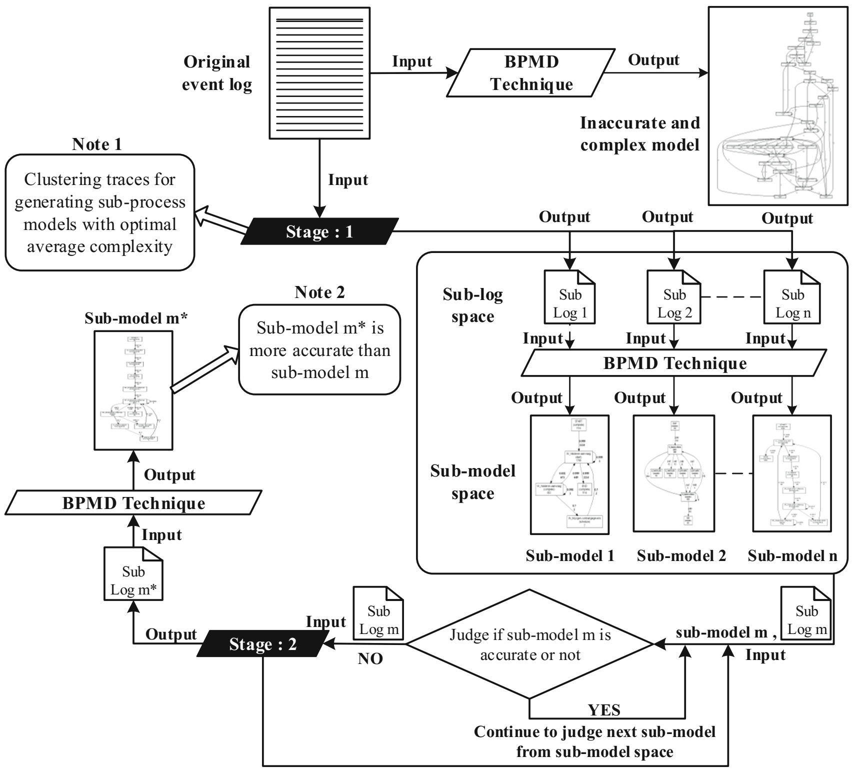


Fig. 2. Outline of the basic idea for the proposed trace clustering technique CTC.

4.1 Stage 1: Trace Clustering

For the first stage of our approach, we developed a new trace clustering method, referred to as *top-down trace clustering* (TDTC). The main idea of our method is to convert the traditional trace clustering problem that is based on a notion of similarity of traces, into a clustering problem that is guided by the complexity of the sub-process models derived for the sub-logs.

Let $\varPhi = \{\phi_1, \phi_2, \ldots, \phi_n\}$ be a solution space, where each solution $\phi_m \in \varPhi$ stands for a unique way to divide the original event log into a fixed number of sub-logs. TDTC employs a greedy strategy to search for the optimal solution ϕ_{op} of $\varPhi$, which is characterised by an optimal weighted average complexity of the sub-process models constructed for the generated sub-logs. As shown in Fig. 3, for a log L and a target number (three in this example) of sub-logs, TDTC first searches for the optimal way to divide L into two sub-logs L_1 and L_2. Then, TDTC continues to detect the optimal way to split L_2 (which is assumed to lead to a sub-model with the highest complexity) into L_3 and L_4. This basic idea is instantiated based on the following concepts related to sequential patterns in traces, henceforth called trace behaviours.

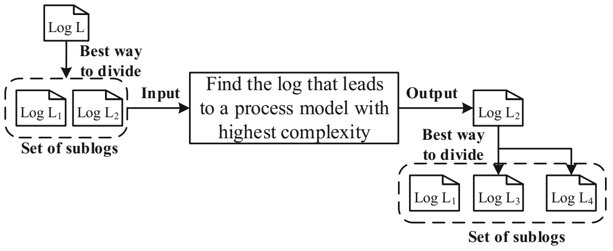


Fig. 3. Illustration of the basic idea for top-down trace clustering.

Significant Trace Behaviours. A complex business process can often be divided into several simpler sub-processes, where each sub-process is characterised by specific behavioural patterns [6]. We refer to the representation of these behavioural patterns in the event log as *trace behaviours*. When conducting trace clustering, we are particularly interested in trace behaviours that adhere to a sub-process model that is simpler than the one that would be discovered for the whole event log. We call these trace behaviours *complexity-related significant behaviours* (CRSB) and detecting them enables us to split up an event log, such that the discovered sub-process models are of low complexity.

We first define trace behaviours in a formal way, based on the notion of sequential patterns as introduced in Sect. 3. That is, a trace behaviour is a sequential pattern mined from a given event log, as the latter can be seen as a database of sequences.

Definition 2 *(Trace Behaviours). Let $\varGamma$ be a closed sequential pattern mining algorithm and min_sup be a minimum support value. Then, the set of trace behaviours $\varTheta$ of an event log L is defined as $\varTheta = \{\theta \mid \theta \in \varGamma(L, min\_sup)\}$.*

The idea behind grounding trace behaviours in sequential patterns is that certain frequent sub-sequences among the traces of an event log are able to reveal some significant criteria about the behavioural patterns in business processes. They may therefore help to distinguish sub-process models that represent different variations of a business process. Moreover, we note that relying on sequential patterns is also in line with the idea of most advanced BPMD algorithms, which cope with noise in the event data by taking the frequency of behavioural patterns into account in the construction of a process model.

As a next step, we classify trace behaviours of an event log into complexity-related significant behaviours (CRSB) and complexity-related insignificant behaviours (CRIB). Let L be an event log; θ be a trace behaviour of L; $L_1 \subseteq L$ be a sub-log of L which contains all the traces with sub-sequence θ from L; $L_2 \subseteq L$ be a sub-log of L which consists of all the traces from L without sub-sequence θ; and $m_1 = |L_1|$ and $m_2 = |L_2|$ be the total numbers of traces in sub-logs L_1 and L_2 respectively. Furthermore, let $v_L = \Sigma(\Lambda(L))$, $v_{L_1} = \Sigma(\Lambda(L_1))$ and $v_{L_2} = \Sigma(\Lambda(L_2))$ be three assessed values generated by implementing the process model complexity evaluation mechanism Σ on the process models for L, L_1 and L_2. Based thereon, we define *sub-model improvement on complexity* $SMI_C(L_1, L_2, L)$ as a measure to quantify the impact of a particular trace behaviour to split the log L into sub-logs L_1 and L_2:

$$SMI_C(L_1, L_2, L) = \frac{(v_L - (m_1 \cdot v_{L_1} + m_2 \cdot v_{L_2})/(m_1 + m_2))}{v_L}. \tag{1}$$

Using this measure, we characterise complexity-related significant behaviours (CRSB) and complexity-related insignificant behaviours (CRIB). That is, a trace behaviour θ is judged to be a CRSB, if it is able to divide the original event log L into two sub-logs, such that the weighted average complexity of the sub-models discovered from the sub-logs can be decreased by at least η, in comparison to the complexity of the model discovered for the original event log.

Definition 3 *(CRSB and CRIB). Given a minimum threshold η, a trace behaviour $\theta \in \Theta_L$ is a complexity-related significant behaviour, if $SMI_C(L_1, L_2, L) \geq \eta$, otherwise θ is a complexity-related insignificant behaviour.*

Top-Down Trace Clustering (TDTC). Using the above notions, Algorithm 1 describes our top-down trace clustering method. TDTC applies a greedy strategy, which detects the best CRSB for iteratively splitting the event log. According to Algorithm 1, for an input event log L, TDTC first acquires the set of trace behaviours TB for L and initialises the set of logs SL (line 1). Afterwards, TDTC iteratively divides the log L into several sub-logs until the total number of generated sub-logs reaches μ or no log in SL can be further divided (lines 2–9). As shown in line 6, if the found trace behaviour tb_m is not a CRSB, then it will not be utilised for dividing the log. This means that, if the average complexity of the sub-process models discovered from the generated sub-logs (i.e., L_{n1} and L_{n2}) cannot be decreased to a certain extent compared to the quality of the model discovered from the original event log (i.e., L_n), then it is not worth splitting the log. Intuitively, this requirement is derived from the goal to achieve a balance between the integrity and the quality of the resulting models. Additionally, if the number of traces in the generated sub-logs (i.e., L_{n1} and L_{n2}) is less than threshold κ, then the found trace behaviour tb_m will also not be used for splitting. Here, threshold κ is used to prevent TDTC from generating sub-logs with too few traces. Finally, an array of sub-logs SL is returned by TDTC.

Algorithm 1. Top-down trace clustering (TDTC)

```
Input: an event log $L$, a minimum support $min\_sup$ for mining closed sequential
    patterns, a minimum threshold $\eta$ for detecting CRSB, the minimum size $\kappa$ for each
    generated sub-log, the target number of generated sub-logs $\mu$.
    Let $TB$ be a set of trace behaviours.
    Let $SL$ be a set of event log.
1: $TB \leftarrow \Gamma(L, min\_sup)$, $SL \leftarrow SL \cup L$
2: repeat
3:    find the log $L_n \in SL$ which leads to a model with the highest complexity
4:    find the trace behaviour $tb_m \in TB$ to generate the highest $SMI_C$ for log $L_n$
5:    split log $L_n$ into $L_{n1}$ and $L_{n2}$ by employing trace behaviour $tb_m$
6:    if $SMI_C(L_{n1}, L_{n2}, L_n) \geq \eta$ and $|L_{n1}| \geq \kappa$ and $|L_{n2}| \geq \kappa$ then
7:       remove $L_n$ from $SL$ and put $L_{n1}$ and $L_{n2}$ in $SL$
8:    end if
9: until (no log in $SL$ can be further divided or the cluster number $\mu$ is reached)
Output: a set of event logs $SL$.
```

4.2 Stage 2: Process Model Fitness Improvement

As part of our compound trace clustering technique, the accuracy of the sub-process models stemming from stage 1 is improved in a second stage (see Fig. 2). In particular, we consider fitness [13] as a well-established measure for the accuracy in process model discovery. Specifically, we employ a fitness improvement algorithm named HIF [17] and apply it to each of the sub-process models. In essence, HIF locates behavioural patterns recorded in the event log, which cannot be expressed by the utilised BPMD algorithm. It then converts these patterns into behavioural structures that can be expressed by the discovery algorithm, so that a more fitting process model will be obtained.

4.3 The Compound Trace Clustering (CTC) Algorithm

Putting the above techniques together, the complete approach of compound trace clustering for process discovery is formalised in Algorithm 2. In addition to the above notions, this algorithm relies on a process model fitness evaluation measure $\Delta : (M^+, L^+) \rightarrow \mathbb{R}$, where M^+ is the universe of process models and L^+ is the universe of event logs.

As described above before, CTC contains two stages. In stage 1, TDTC (introduced in Algorithm 1) is employed to divide the original event log L into a fixed number (indicated by parameter μ) of sub-logs, which are then stored in set SL (line 2 of Algorithm 2). In stage 2, if a sub-log sl from SL leads to a sub-process model with a fitness value less than a given target value ϵ (line 4), then HIF is used to transform the respective sub-log until the discovered sub-process model has a fitness value of no less than ϵ (line 5). Finally, the sub-process models with improved fitness are stored in MO (lines 6 and 8), which forms the output of CTC. Note that the time complexity of CTC depends on the chosen algorithms for closed sequential pattern mining (Γ) and BPMD (Λ).

Algorithm 2. The compound trace clustering technique: CTC

```
Input: an event log $L$, a minimum support $min\_sup$ for mining closed sequential
    patterns, a minimum threshold $\eta$ for detecting CRSB, the minimum size $\kappa$ for each
    generated sub-log, the target number of generated sub-logs $\mu$, a target fitness value
    $\epsilon$ for the sub-process model.
    Let $SL$ be an array of event log.
    Let $MO$ be a set of sub-process models.
 1: $SL \leftarrow null$, $MO \leftarrow null$
    Stage 1: cluster traces for generating sub-process models with optimal complexity
 2: $SL \leftarrow TDTC(L, min\_sup, \eta, \kappa, \mu)$
    Stage 2: generate high-fitness sub-process models
 3: for each sub-log $sl \in SL$ do
 4:     if $\Delta(\Lambda(sl), sl) < \epsilon$ then
 5:         $sl \leftarrow HIF(sl, \epsilon)$
 6:         $MO \leftarrow MO \cup \Lambda(sl)$
 7:     else
 8:         $MO \leftarrow MO \cup \Lambda(sl)$
 9:     end if
10: end for
Output: a set of sub-process models $MO$, an array of event log $SL$.
```

5 Evaluation

This section presents an experimental evaluation of the proposed method of compound trace clustering for process discovery. We first review the used datasets and experimental setup, before turning to a discussion of the obtained results.

Datasets. We tested the effectiveness of CTC on four real-life event logs: an event log of a Volvo IT incident and problem management process (VIPM) published as part of the Business Process Intelligence Challenge (BPIC) 2013; a log of a loan and overdraft approvals process (LOA) of BPIC 2012; a log of an ICT service process (KIM); and a log of a CRM process (MCRM) from [6]. Descriptive statistics of these event logs are given in Table 2.

Experimental setup. To evaluate the quality of the discovered models, a process model complexity measure is used. To this end, we exploit the insights reported in [14], which highlight that the density, the number of control-flows arcs, and the number of model elements are the main factors that influence the

Table 2. Basic information of the evaluated logs.

Log	Traces	Events	Event types
VIPM	7554	65533	13
LOA	13087	262200	36
KIM	24770	124217	18
MCRM	956	11218	22

comprehensibility of a process model that is expressed as a Petri-net [1]. More specifically, we rely on the Place/Transition Connection Degree (PT-CD) metric for quantifying complexity of a Petri-net, see [6]. With $|ar|$ as the total number of arcs in the model, $|P|$ as the number of places, and $|T|$ as the number of transitions, the PT-CD is defined as:

$$PT - CD = \frac{1}{2}\frac{|ar|}{|P|} + \frac{1}{2}\frac{|ar|}{|T|} \tag{2}$$

Here, large values of the PT-CD metric indicate a high complexity of the model. As an alternative measure for model complexity, we further consider the Extended Cardoso metric (E-Cardoso) [19]. It quantifies the control flow complexity of process models. A higher E-Cardoso value indicates a more complex model.

In our experiments, we further use the Flexible Heuristics Miner (FHM) [11], as implemented in ProM 6[1] as the business process model discovery algorithm. This choice is motivated by the algorithm's robustness against noise and its computational efficiency. Since FHM constructs a process model that is given as a Heuristics Net, we rely on the *Heuristics Net to Petri Net* plugin in ProM 6 to convert the result of FHM into a Petri-net. The complexity of this Petri-net is then assessed based on the aforementioned measures.

To assess the accuracy of the discovered models, we rely on the ICS fitness measure [13], which falls into rage $(-\infty, 1]$ and can be computed efficiently. In addition, we consider the *F-score*, which is defined as the harmonic mean of recall (fitness) and precision (appropriateness) [18]. To quantify precision of the discovered sub-process models, we utilise the ETConformance Checker as it is implemented in ProM 6.

When running CTC, the minimum support value min_sup for closed sequential pattern mining is set to 0.1 for the logs VIPM, KIM, and MCRM; and to 0.25 for log LOA. The reason being that the first three logs contain less process variants compared to LOA. The minimum threshold η for detecting CRSB is set to 0 (i.e., condition $SMI_C > 0$ should be fulfilled), while the minimum size κ for each sub-log is set to 50. The target number of generated sub-logs μ is varied in the experiments up to a value of 6. The target fitness value ϵ for each sub-process model is set to 1.

[1] http://www.promtools.org.

Results. A first overview of our evaluation results (when μ is set to 5) is shown in Table 3. For each measure and log, Table 3 first gives the obtained value for the sub-process models obtained by CTC (averaged over all sub-process models), before also listing the value for the model discovered from the original event log. For instance, the weighted average ICS fitness of the sub-process models obtained with CTC on the log VIPM is 0.9159 while the ICS fitness of the model discovered from the original event log is 0.3594.

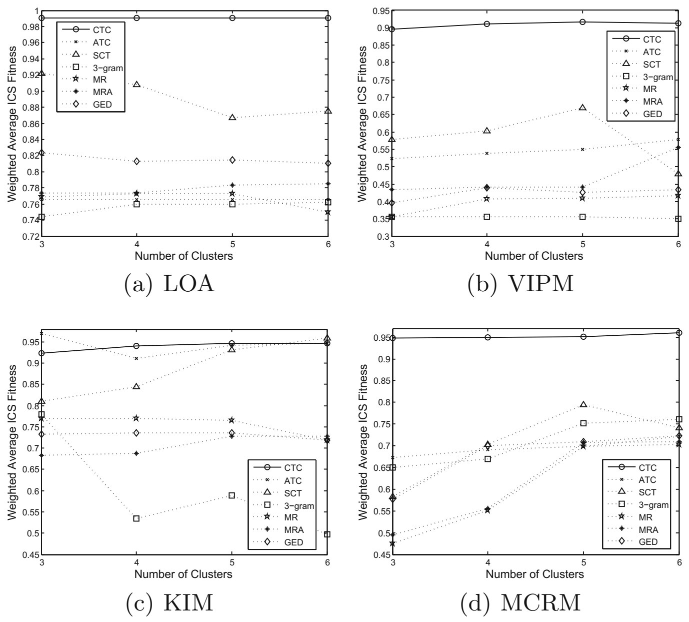
The evaluation results shown in Table 3 highlight that the weighted average fitness of the generated sub-models for each event log is much higher than the fitness of the model discovered from the original log, whereas the average complexity of these sub-models is relatively low. As such, the results demonstrate the effectiveness of our approach to compound trace clustering for process discovery.

Table 3. Evaluation results for the sub-models generated by CTC. First values are the average over all sub-process models, whereas the second values are those obtained for the model discovered from the original event log.

Event log	Weighted average ICS fitness	Weighted average PT-CD	Weighted average E-Cardoso
VIPM	0.9159/0.3594	2.3577/2.8848	47.3313/54
LOA	0.9909/0.7878	2.4845/3.1478	110.7463/148
KIM	0.9461/0.7904	2.8626/3.4797	63.2614/79
MCRM	0.9512/−0.1379	2.2818/2.4545	51.7364/64

We also compared CTC to six traditional trace clustering techniques, specifically 3-gram [2], ATC [6], MR and MRA [3], GED [4] and sequence clustering (SCT) [5]. For each log, we evaluate the trace clustering technique with different numbers of clusters (from 3, 4, 5 and 6). Figure 4 shows the comparison results from the perspective of fitness.The results illustrate that CTC performs much better on event logs LOA, VIPM and MCRM than the other six trace clustering methods. For the log KIM, ATC has better overall performance than CTC because ATC also has a fitness improvement mechanism that is applied to the sub-process models. However, the mechanism provided by CTC seems more stable on the four real-world event logs.

Figure 5 shows the comparison results on F-score. It can be seen that CTC also performs better than the traditional trace clustering techniques on most of the tested logs. Figure 6 highlights the comparison results from the angle of PT-CD. Here, CTC and SCT outperform the other techniques. Figure 7 depicts the comparison results on E-Cardoso, hinting at an average performance of CTC in comparison to the other methods. The main reason is that the fitness improvement method HIF utilised by CTC may decrease the performance of CTC on optimising the complexity (evaluated by E-Cardoso) of the potential sub-models. Nevertheless, we conclude that under a comprehensive assessment, CTC improves beyond the state-of-the-art in trace clustering in the context of process model discovery.



(a) LOA (b) VIPM

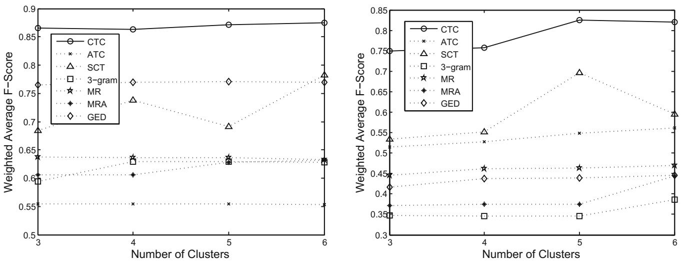
(c) KIM (d) MCRM

Fig. 4. Comparison of weighed average fitness of the sub-models output by the seven trace clustering techniques.

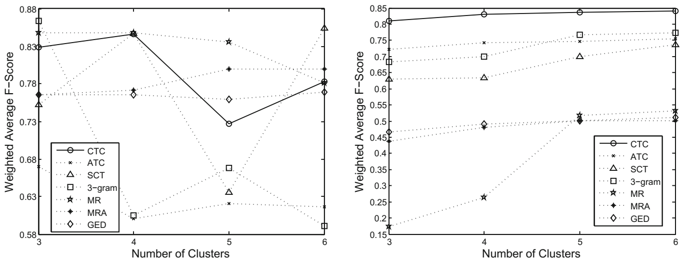
6 Related Work

In the literature, many trace clustering techniques have been put forward to overcome the negative impact of a large variety of complex control-flow patterns recorded in event logs. We classify these proposed techniques into *passive trace clustering methods* and *active trace clustering methods*.

Passive trace clustering methods such as [2–5] try to detect the similarity of traces recorded in event logs and then group the traces with similar structures into the same sub-log. For example, in [2], traces are expressed by profiles. Every profile is a set of items that characterise a trace in terms of a particular aspect. Five profiles, such like the case attributes profile and the event attributes profile, are introduced in [2]. The distance between any two traces is then measured by transforming the defined profiles into an aggregate vector. In [3], the authors pointed out that the feature sets based on repeated sub-sequences of traces are context-aware and able to exhibit some common functionality. The traces that have a lot of common features should be placed in the same cluster. In [4], an

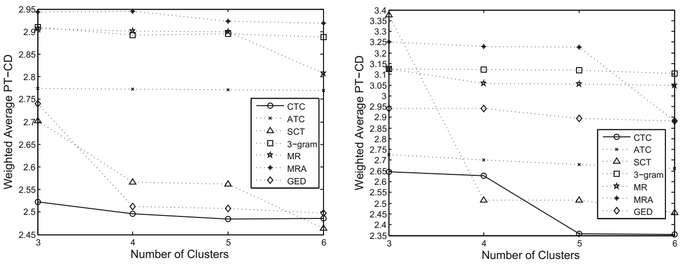


(a) LOA (b) VIPM

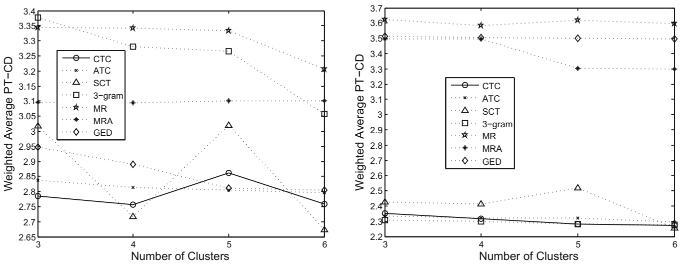


(c) KIM (d) MCRM

Fig. 5. Comparison of weighted average F-score of the sub-models output by the seven trace clustering techniques.



(a) LOA (b) VIPM



(c) KIM (d) MCRM

Fig. 6. Comparison of weighted average PT-CD of the sub-models output by the seven trace clustering techniques.

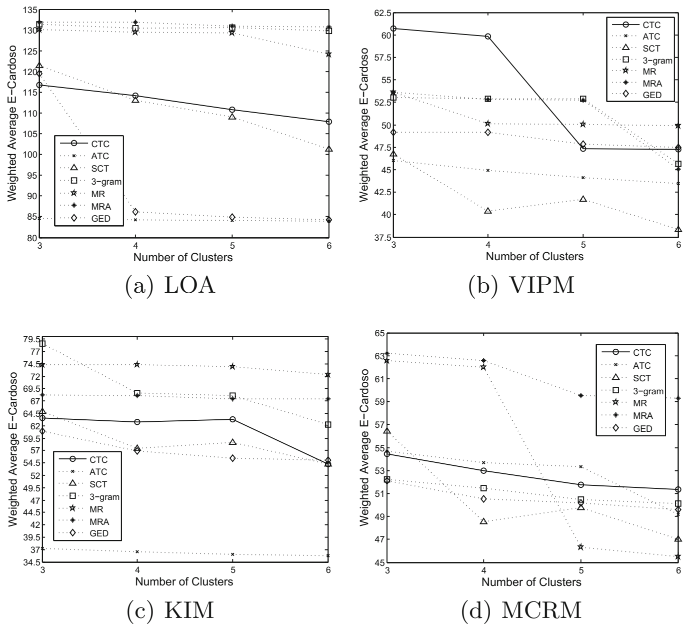


Fig. 7. Comparison of weighted average E-Cardoso of the sub-models output by the seven trace clustering techniques.

edit distance-based approach for trace clustering is proposed. The context-aware knowledge is integrated into the calculation procedure so that the calculated edit distance between any two traces becomes more accurate. In [5], sequence clustering technique is proposed, which learns a first-order Markov model for each cluster. A trace will be put into the cluster that is assigned the Markov model that is able to generate this trace with the highest probability. However, passive trace clustering methods suffer from the gap between the clustering bias and the model evaluation bias [6]. As a result, these techniques cannot ensure the accuracy of the sub-process models constructed from the resulting sub-logs.

Active trace clustering methods such as [6,8–10] assume an integrated view on the clustering bias and the model evaluation bias. For example, ATC as presented in [6], directly optimises the accuracy of the sub-process models derived from sub-logs, similar to CTC proposed in this paper. However, as demonstrated in our experimental evaluation, the mechanism provided by ATC turns out to be not very stable. In contrast, CTC achieves the best results under a comprehensive assessment, when compared to existing active trace clustering methods.

7 Conclusions

In this paper, we proposed a new trace clustering technique named CTC to generate accurate and simple sub-process models. Our technique consists of two stages. In a first stage, it generates sub-process models while striving for an optimal average complexity of the resulting models. In a second stage, the accuracy of the resulting models is improved. Our experimental results demonstrated the effectiveness of our technique, also in comparison to six traditional trace clustering techniques.

In future work, we will focus on improving the performance of CTC by developing new methods to filter trivial trace behaviours found by CTC from real-life event logs. Also, techniques that help to explore the parameter spaces in the configuration of our technique (such as the minimum threshold to detect complexity-related significant behaviours or the minimum size per sub-log) will be explored. Furthermore, we plan to conduct further evaluation studies, validating our methods in additional application domains.

References

1. van der Aalst, W.M.P.: Process Mining: Data Science in Action. Springer, Berlin (2016)
2. Song, M., Günther, C.W., van der Aalst, W.M.P.: Trace clustering in process mining. In: Ardagna, D., Mecella, M., Yang, J. (eds.) BPM 2008. LNBIP, vol. 17, pp. 109–120. Springer, Heidelberg (2009). doi:10.1007/978-3-642-00328-8_11
3. Bose, R.P.J.C., van der Aalst, W.M.P.: Trace clustering based on conserved patterns: towards achieving better process models. In: Rinderle-Ma, S., Sadiq, S., Leymann, F. (eds.) BPM 2009. LNBIP, vol. 43, pp. 170–181. Springer, Heidelberg (2010). doi:10.1007/978-3-642-12186-9_16
4. Bose, R., van der Aalst, W.M.P.: Context aware trace clustering: towards improving process mining results. In: SIAM International Conference on Data Mining, pp. 401–402 (2009)
5. Ferreira, D., Zacarias, M., Malheiros, M., Ferreira, P.: Approaching process mining with sequence clustering: experiments and findings. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 360–374. Springer, Heidelberg (2007). doi:10.1007/978-3-540-75183-0_26
6. Weerdt, J.D., vanden Broucke, S., Vanthienen, J., Baesens, B.: Active trace clustering for improved process discovery. IEEE Trans. Knowl. Data Eng. **25**(12), 2708–2720 (2013)
7. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs - a constructive approach. In: Colom, J.-M., Desel, J. (eds.) PETRI NETS 2013. LNCS, vol. 7927, pp. 311–329. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38697-8_17
8. Ekanayake, C.C., Dumas, M., García-Bañuelos, L., La Rosa, M.: Slice, mine and dice: complexity-aware automated discovery of business process models. In: Daniel, F., Wang, J., Weber, B. (eds.) BPM 2013. LNCS, vol. 8094, pp. 49–64. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40176-3_6
9. Garcia, L., Dumas, M., Rosa, M.L., Weerdt, J.D., Ekanayake, C.C.: Controlled automated discovery of collections of business process models. Inf. Syst. **46**, 85–101 (2014)
10. Greco, G., Guzzo, A., Pontieri, L.: Discovering expressive process models by clustering log traces. IEEE Trans. Knowl. Data Eng. **18**(8), 1010–1027 (2006)

11. Weijters, A.J.M.M., Ribeiro, J.T.S.: Flexible Heuristics Miner (FHM). BETA Working Paper Series, WP 334. Eindhoven University of Technology, Eindhoven (2010)
12. Adriansyah, A., Buijs, J.C.A.M.: Mining process performance from event logs. In: La Rosa, M., Soffer, P. (eds.) BPM 2012. LNBIP, vol. 132, pp. 217–218. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36285-9_23
13. de Medeiros, A.A.: Genetic process mining. Ph.D. thesis, Eindhoven University of Technology (2006)
14. Mendling, J., Strembeck, M.: Influence factors of understanding business process models. In: Abramowicz, W., Fensel, D. (eds.) BIS 2008. LNBIP, vol. 7, pp. 142–153. Springer, Heidelberg (2008). doi:10.1007/978-3-540-79396-0_13
15. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann, San Francisco (2000)
16. Shengnan, C., Han, J., David, P.: Parallel mining of closed sequential patterns. In: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, KDD 2005, pp. 562–567. ACM, New York (2005)
17. Sun, Y., Bauer, B.: A novel heuristic method for improving the fitness of mined business process models. In: Sheng, Q.Z., Stroulia, E., Tata, S., Bhiri, S. (eds.) ICSOC 2016. LNCS, vol. 9936, pp. 537–546. Springer, Cham (2016). doi:10.1007/978-3-319-46295-0_33
18. Conforti, R., Dumas, M., García-Bañuelos, L., La Rosa, M.: Beyond tasks and gateways: discovering BPMN models with subprocesses, boundary events and activity markers. In: Sadiq, S., Soffer, P., Völzer, H. (eds.) BPM 2014. LNCS, vol. 8659, pp. 101–117. Springer, Cham (2014). doi:10.1007/978-3-319-10172-9_7
19. Lassen, K.B., van der Aalst, W.M.P.: Complexity metrics for workflow nets. Inf. Softw. Technol. **51**(3), 610–626 (2009)