A Novel Top-Down Approach for Clustering Traces

Yaguang Sun(✉) and Bernhard Bauer

Programming Distributed Systems Lab, University of Augsburg, Augsburg, Germany
{yaguang.sun,bernhard.bauer}@informatik.uni-augsburg.de

Abstract. In the last years workflow discovery has become an important research topic in the business process mining area. However, existing workflow discovery techniques encounter challenges while dealing with event logs stemming from highly flexible environments because such logs contain many different behaviors. As a result, inaccurate and complex process models might be obtained. In this paper we propose a new technique which searches for the optimal way for clustering traces among all of the possible solutions. By applying the existing workflow discovery techniques on the traces for each discovered cluster by our method, more accurate and simpler sub-models can be obtained.

Keywords: Business process mining · Trace clustering · Greedy algorithm · Business process extension

1 Introduction

Business process mining techniques aim at discovering, monitoring and improving real processes by extracting knowledge from event logs recorded by enterprise information systems [1]. In general, current process mining techniques mainly consider three perspectives: workflow discovery, conformance checking and process extension [2]. The starting point of these analyses is usually an event log which is a set of cases, where each case is an instance of a business process. Every case in an event log has an attribute *trace* which is a set of ordered events. Cases and events are uniquely identified in the event log by *case id* and *event id* respectively. Additionally, typical event logs may contain much more process information, e.g., the performer and cost of each event.

However, in the real world many business processes are often executed in highly flexible environments, e.g., healthcare, customer relationship management (CRM) and product development [3]. As a result, the existing business process mining techniques might generate inaccurate and impalpable analysis results while dealing with event logs (real-life logs) stemming from such flexible environments. The problem is largely due to the dense distribution of cases with a high variety of behaviors in the real-life event log.

As one of the most crucial learning task in the business process mining area, the current workflow discovery techniques also encounter great challenges in the

J. Zdravkovic et al. (Eds.): CAiSE 2015, LNCS 9097, pp. 331–345, 2015.
DOI: 10.1007/978-3-319-19069-3_21

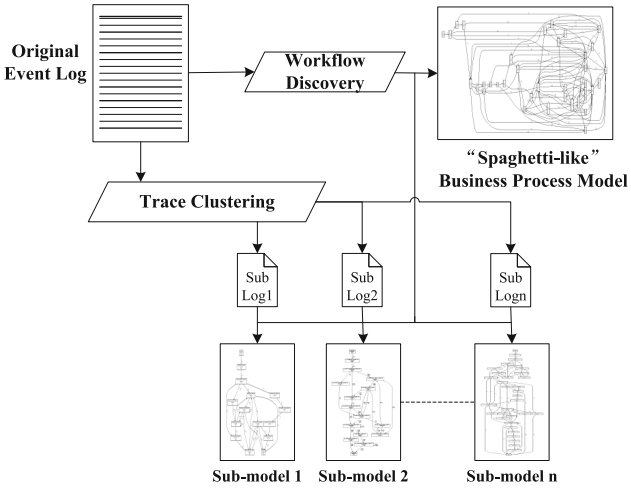


Fig. 1. Illustration of the basic trace clustering procedure in process mining

scenario of real-life event logs. For instance, "spaghetti-like" business process models might be generated by existing process discovery algorithms with an input of real-life event log [2]. Such models are often inaccurate and too complex to be well interpreted. Accordingly, some pioneering approaches have been developed to solve this problem. One efficient technique is *trace clustering* [3–7] which mines the structural behaviors[1] of traces (trace behaviors) in an event log and then groups the traces with similar behaviors into the same sub-log. Afterwards, by applying workflow discovery algorithms on each simpler sub-log, more accurate and comprehensible process models can be obtained. Figure 1 shows the basic procedure for *trace clustering*.

Nevertheless, most currently available *trace clustering* techniques treat all of the trace behaviors captured in the event log equally. As a result, the impacts of some important trace behaviors are reduced. Moreover, these techniques focus mainly on the discovery of various kinds of trace behaviors while the quality of the underlying process model for each cluster learned is not taken into account [3]. Hence, high-quality sub-process models from these trace clustering techniques can not be guaranteed. A promising method called *Active Trace Clustering* (ATC) was put forward in [3] which directly optimises the accuracy of each cluster's underlying process model. However, ATC only considers model accuracy metrics while the complexity of process models is neglected during trace clustering. The complexity of process models is also a very important metric and should not be ignored for *trace clustering*. Because a highly accurate process model can still be very complicated.

[1] Most trace clustering techniques only consider the structural behaviors of traces, while some consider the behaviors from both traces and other case attributes.

In this paper, the trace clustering problem is surveyed from a new perspective and redefined as an issue of searching for a global optimal solution in a solution space. The proposed technique employs a greedy strategy for searching for the optimal way to cluster the traces in an event log based on a specific model evaluation schema that considers both the accuracy and complexity of the potential sub-process models during the run time:

- The problem addressed by this paper is discussed in Section 2.
- Section 3.2 formalises definitions related to trace behaviors firstly. Afterwards, four different kinds of trace behaviors are defined for helping cluster the traces.
- In Section 3.3, a top-down approach is put forward which identifies the optimal solution for the trace clustering problem.
- To test the efficiency of our method, we carry out a case study in Section 4 by applying our approach to a real-life event log of the loan and overdraft approvals process from Business Process Intelligence Challenge 2012 (BPIC 2012).

2 Problem Description

*U*nder certain conditions, an inaccurate and complex business process can be divided into several simpler and more accurate sub-processes where each sub-process performs some unique functions reflected by certain specific sub-process constructional behaviors. These behaviors can be recorded in the event log after the execution of the sub-process and expressed through the structural behaviors of traces (trace behaviors). In this paper, the trace behaviors that adhere to a more accurate and simpler sub-process model compared with the original model (generated by using the original event log) are called significant behaviors (defined in Section 3.2). Discovering these significant trace behaviors from the event log will assist in mining better sub-process models by clustering the traces based on these behaviors. However, due to the lack of domain knowledge about the significant trace behaviors, capturing them directly from the event log seems to be a difficult task.

In this paper, we transform the traditional trace clustering problem into the problem of finding the optimal way for clustering the traces among all possible solutions. As shown in Figure 2, each element in the solution space represents one strategy for clustering the traces from an event log into several subsets of traces. A best solution is defined as a solution which is able to divide the traces in the original event log into several subsets where the overall quality of the underlying sub-models for these subsets is optimal. Given a process model evaluation schema, how to find the optimal solution for clustering the traces from an event log is the main problem that this paper is going to solve.

In this paper, we propose a new technique which inherits the basic ideas of *traditional trace clustering techniques* and ATC for discovering the optimal way of clustering the traces. This technique considers both the behaviors of traces and the accuracy and complexity of each potential sub-process model during the mining procedure for the optimal solution.

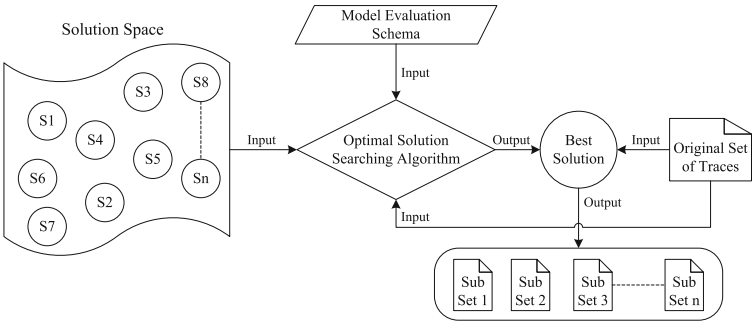


Fig. 2. The process for searching for the optimal solution for clustering traces

3 Approach Design

In this section we propose a new trace clustering technique which differs from existing correlative techniques because it searches for an optimal way for clustering the traces among all of the possible solutions. Four kinds of trace behaviors defined in Section 3.2 provide a basis for this technique to carry out the searching process.

3.1 Notation

Before introducing our method, we discuss some of the important basic concepts and notations. Let I be the set of all items, S be the set of all finite sequences over I. A sequence $S_j \in S$ of length n is denoted $< i_{j1}, i_{j2}, \ldots, i_{jn} >$, where each item i_{jk} represents an item from I. For any two sequences $\alpha =< a_1, a_2, \ldots, a_l >$ and $\beta =< b_1, b_2, \ldots, b_q >$ from S, α is a subsequence of β, denoted as $\alpha \sqsubseteq \beta$, if $1 \leq p_1 < p_2 < \cdots < p_l \leq q$ such that $a_1 = b_{p_1}, a_2 = b_{p_2}, \ldots, a_l = b_{p_l}$.

Let $\mathcal{SC}$ be the set of event logs, $\mathcal{ST}$ be the set of all sets of traces from $\mathcal{SC}$, $\Omega : \mathcal{ST} \rightarrow \mathcal{SY}$ be a workflow discovery algorithm, where $\mathcal{SY}$ is the set of process models. $\Sigma : (\mathcal{SY}, \mathcal{ST}) \rightarrow \mathcal{SV}$ represents a process model evaluation schema with an input of process model and a set of traces and an output of assessed value from $\mathcal{SV}$ (the set of all possible values output by Σ).

Let D be a database of sequences, for a given minimum support min_sup $(0 < min\_sup < 1)$, a sequence λ is called a *sequential pattern* if $support(\lambda) \geq min\_sup \cdot |D|$, where $support(\lambda)$ is the number of sequences in D which contain λ and $|D|$ represents the total number of sequences in D. The set of *sequential patterns*, SP, contains all of the subsequences from D whose support values are no less than min_sup. The set of *closed sequential patterns* is defined as $CSP = \{\alpha | \alpha \in SP \, and \nexists \beta \in SP \, such \, that \, \alpha \sqsubseteq \beta \, and \, support(\alpha) = support(\beta)\}$. $\Gamma : \mathcal{SD} \overset{smin\_sup}{\longrightarrow} \mathcal{SCSP}$ represents a closed sequential pattern mining algorithm, where $\mathcal{SD}$ is the set of all databases of sequences, $\mathcal{SCSP}$ is the set of all sets of closed sequential patterns and $smin\_sup$ is the set of all possible minimum

supports. *CSP* effectively decreases the total number of sequential patterns generated but in the meantime preserves the complete information about all the sequential patterns. Additional information related to *sequential pattern mining* techniques can be found in [8,9].

In [10], the authors make some pioneering researches on identifying the factors that influence the comprehensibility of a business process model expressed as Petri net [11]. These factors primarily include the number of control-flows and the number of components (such as *and-joins*, *and-splits*, *xor-joins*, *xor-splits*, arcs, places and transitions, etc.) in the process model. Based on this previous research the authors of [3] develop an effective metric called *Place/Transition Connection Degree* (PT-CD) for quantifying the complexity of a Petri net. Let $|a|$ be the total number of arcs in the process model, $|P|$ be the number of places and $|T|$ be the number of transitions, the PT-CD is defined as [3]:

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

The greater the PT-CD is, the more complicated the model will be. In this paper we employ the *Heuristics Miner* (HM) [12] for generating the process models because HM is well designed to deal with real-life event logs and also has a good computational performance. Then the *Heuristic Net to Petri Net* plugin in ProM[2] is used for transforming the heuristic net output by the HM into a Petri Net. Afterwards, the PT-CD is used for evaluating the complexity of the Petri Net obtained.

3.2 Concepts Related to Trace Behaviors

Traces are generated performing a specific category of functions determined by business process-based domain criterion. Such criteria can be very diverse, e.g., presence or absence of activities, presence or absence of combinations of activities [3]. These underlying criteria are recorded in the event log and reflected by certain compositional behaviors of traces (trace behaviors). In the first step our technique searches for the structural behaviors of traces in an event log, then according to the identified trace behaviors the optimal solution searching process is carried out.

Trace Behaviors and Significant Trace Behaviors. Given a closed sequential pattern mining algorithm Γ and a minimum support min_sup, the set of trace behaviors TB from an event log C is defined as:

Definition 1. $TB = \{tb | tb \in \Gamma(T_C, min\_sup)\}$, where T_C is the set of traces from C.

According to Definition 1, a trace behavior tb is equivalent to a frequent pattern mined from T_C. In our opinion, certain frequently appeared subsequences

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

among traces in an event log are able to reveal some particularly important criteria of business processes and can help distinguish sub-process models with different functions hidden in the event log. Another benefit of utilising sequential patterns is that they can not only represent consecutive structural behaviors of traces, but inconsecutive trace behaviors as well. For instance, given an event log C_e that contains a set of traces $T_{C_e} = \{< A, C, D, E >, < A, C >, < A, E >\}$ and a minimum support $min\_sup = 0.4$, the set of trace behaviors $TB = \{< A >, < A, C >, < A, E >\}$ can be discovered, the sequential pattern $< A, C >$ is a consecutive trace behavior because activity C always appears right next to A in a trace, and $< A, E >$ is an inconsecutive trace behavior because activity A and E may appear in a trace discretely. However, most existing pattern-based trace clustering techniques are only able to capture consecutive trace behaviors in an event log. Moreover, employing frequent patterns is also in accordance with the main idea of most advanced process discovery techniques: only the frequent structures should be considered in the process mining procedure [2].

Additionally, we classify the behaviors of traces from a real-life event log into *significant behaviors* and *nonsignificant behaviors*. Let T_C be a set of traces from the event log C, tb is a trace behavior discovered from C, $C_1 \subseteq C$ is the sub-log of C, where T_{C_1} consists of all of the traces with a subsequence tb, $C_2 \subseteq C$ is the sub-log of C where T_{C_2} contains all of the traces without a subsequence tb, $V_C = \Sigma(\Omega(T_C))$, $V_{C_1} = \Sigma(\Omega(T_{C_1}))$ and $V_{C_2} = \Sigma(\Omega(T_{C_2}))$ are the assessed values obtained by performing the process model evaluation schema Σ on the process models for T_C, T_{C_1} and T_{C_2}. The significant behavior is conveyed by the following definition:

Definition 2. For a given minimum threshold μ, the trace behavior $tb \in TB$ is called a *significant trace behavior* (STB) if $((V_{C_1} + V_{C_2})/2 - V_C)/V_C \geqslant \mu$, otherwise tb is called an insignificant behavior.

As stated in Definition 2, a STB is able to divide the original set of traces into two subsets that lead to two process models of which the average quality should be increased by at least μ (a minimum threshold) compared with the quality of the model generated by utilising the original set of traces.

Sub-Model Improvement for STB. According to Definition 2, the starting point for identifying a STB is a process model evaluation schema Σ. As mentioned in Section 2, while evaluating a process model both the accuracy and complexity should be taken into account. Accordingly, the model evaluation schema Σ should contain two parts: the fitness[3] [2] computing schema Σ_f and the complexity evaluation schema Σ_c. Let Ω be a process model mining algorithm, T_C be a set of traces from the event log C, a trace behavior tb discovered from C separates T_C into T_{C_1} and T_{C_2}, where C_1 and C_2 stand for the sub-logs of C, the sub-model improvement SMI is defined as:

[3] Fitness is an important metric for calculating the accuracy of a process model which quantifies how good the behaviors in the event log can be expressed by the model.

$$SMI(T_{C_1}, T_{C_2}, T_C) = \alpha \times SMI_F(T_{C_1}, T_{C_2}, T_C) + \beta \times SMI_C(T_{C_1}, T_{C_2}, T_C) \quad (2)$$

$$SMI_F(T_{C_1}, T_{C_2}, T_C) = \frac{\frac{1}{2}(\Sigma_f(\Omega(T_{C_1})) + \Sigma_f(\Omega(T_{C_2}))) - \Sigma_f(\Omega(T_C))}{\Sigma_f(\Omega(T_C))} \quad (3)$$

$$SMI_C(T_{C_1}, T_{C_2}, T_C) = \frac{\Sigma_c(\Omega(T_C)) - \frac{1}{2}(\Sigma_c(\Omega(T_{C_1})) + \Sigma_c(\Omega(T_{C_2})))}{\Sigma_c(\Omega(T_C))} \quad (4)$$

According to Equation 2, the SMI is composed by two parts. The first part is related to the model accuracy and the second part is related to the model complexity. Our technique utilises the ICS fitness [13] and PT-CD for the evaluation of process models. The main reason for using the ICS fitness is that it has a computationally efficient calculative process and also includes a punishment schema for an underfitting process model (such models allow for many additional behaviors that are not registered in the event logs). In Equation 2, α and β represent the weights for the two parts and meet the condition of $\alpha+\beta=1$. The values of α and β should be set upon the conditions of accuracy and complexity of the original model. For instance, if the original model has a good accuracy but suffers from a bad complexity then the value of β should be set higher than α and vice versa. According to Definition 2, given a minimum threshold μ, the trace behavior *tb* is a STB if $SMI(T_{C_1}, T_{C_2}, T_C) \geq \mu$.

Strict STB and Conditional Strict STB. The sub-model improvement criterion SMI considers both the fitness and complexity of the process models at the same time. However, in reality the fitness and complexity of a model are not associated with each other. The increment of fitness is not always accompanied by a decrement of the model complexity and vice versa. For example, let *tb* be a trace behavior from the event log C which divides the original set of traces T_C into T_{C_1} and T_{C_2}, pretend that $\mu = 0.15$, $\alpha = 0.5$, $\beta = 0.5$, $SMI_F(T_{C_1}, T_{C_2}, T_C) = -0.1$, $SMI_C(T_{C_1}, T_{C_2}, T_C) = 0.4$, according to Equation 2 and Definition 2, the $SMI(T_{C_1}, T_{C_2}, T_C) = 0.15$ is equal to the value of μ so *tb* is judged to be a STB. Even though the average fitness of the sub-models for T_{C_1} and T_{C_2} is decreased, the value of $SMI(T_{C_1}, T_{C_2}, T_C)$ augments because the average complexity of the sub-models is greatly reduced.

To avoid this situation, a stricter definition for STB needs to be developed. Let *stb* be a STB mined from a log C which divides the set of traces T_C into T_{C_1} and T_{C_2}, the *strict significant trace behavior* is defined as follows:

Definition 3. The *stb* is called a *strict significant trace behavior* (SSTB) if $SMI_F(T_{C_1}, T_{C_2}, T_C) \geq \mu_f$ and $SMI_C(T_{C_1}, T_{C_2}, T_C) \geq \mu_c$, where μ_f is a minimum threshold for the average fitness increment of the models for T_{C_1} and T_{C_2} compared with the original model and μ_c is a minimum threshold for the average complexity decrement of the models for T_{C_1} and T_{C_2}.

Based on Definition 3, a SSTB satisfies all the conditions for STB, in the meantime some additional conditions should be fulfilled: both the average fitness and average complexity of the related sub-models need to be improved to a certain extent.

Let Σ_f be a fitness computing schema, Σ_c be a complexity computing schema, Ω be a process model mining algorithm, given a minimum threshold φ_f and a maximum threshold φ_c:

Definition 4. The trace behavior tb is called a fitness-based conditional strict STB (FCSTB) if $(\Sigma_f(\Omega(T_{C_1})) + \Sigma_f(\Omega(T_{C_2})))/2 \geq \varphi_f$, $(SMI_C(T_{C_1}, T_{C_2}, T_C) \geq \mu_c \vee (\Sigma_c(\Omega(T_{C_1})) + \Sigma_c(\Omega(T_{C_2})))/2 \leq \varphi_c)$ and $SMI(T_{C_1}, T_{C_2}, T_C) \geq \mu$.

Definition 5. The trace behavior tb is called a complexity-based conditional strict STB (CCSTB) if $(\Sigma_c(\Omega(T_{C_1}))+\Sigma_c(\Omega(T_{C_2})))/2 \leq \varphi_c$, $(SMI_f(T_{C_1}, T_{C_2}, T_C) \geq \mu_f \vee (\Sigma_f(\Omega(T_{C_1})) + \Sigma_f(\Omega(T_{C_2})))/2 \geq \varphi_f)$ and $SMI(T_{C_1}, T_{C_2}, T_C) \geq \mu$.

The FCSTB is defined to deal with an event log of which the potential model has a high fitness but an inferior complexity. For instance, let tb be a trace behavior from the event log C which divides the original set of traces T_C into T_{C_1} and T_{C_2}, pretend that $\mu = 0.15$, $\alpha = 0.5$, $\beta = 0.5$, $SMI_F(T_{C_1}, T_{C_2}, T_C) = -0.1$, $SMI_C(T_{C_1}, T_C) = 0.4$, $\varphi_f = 0.9$, $(\Sigma_f(\Omega(T_{C_1})) + \Sigma_f(\Omega(T_{C_2})))/2 = 0.93$, according to Definition 2 and Definition 3, tb is a STB but not a SSTB. However, even though the average fitness of the sub-models decreases compared to the original model, it still remains a large value and greater than φ_f. In such a situation, the effect of tb should not be neglected. A corresponding definition to FCSTB is complexity-based conditional strict STB (CCSTB) which is defined in Definition 5. It should also be noticed that a tb can be both the FCSTB and the CCSTB at the same time.

3.3 A Top-Down Algorithm for Clustering Traces

In this section an algorithm is put forward for finding the optimal way to cluster the traces in an event log based on the definitions elaborated in Section 3.2. This algorithm applies a greedy strategy which discovers the best trace behavior (that is either a SSTB or a FCSTB or a CCSTB) for splitting the original set of traces for each stage according to the value of SMI. Let $\Pi(TB, T, \theta)$ be a trace behavior removing method, TB represents a set of trace behaviors mined from the set of traces T, a trace behavior $tb \in TB$ is able to divide T into two subsets: T_1 (contains the traces with a subsequence tb) and T_2 (contains the traces without a subsequence tb), if $|T_1| \leq \theta$ or $|T_2| \leq \theta$ then tb is removed from TB. In our technique θ stands for a minimum number of traces for each cluster. A trace behavior that leads to a cluster with a number of traces less than θ will not be considered. Given a workflow discovery algorithm Ω, a closed sequential pattern mining algorithm Γ, a process model fitness evaluation schema Σ_f and a process model complexity evaluation schema Σ_c, the details of our method is described in Algorithm 1.

To prevent the tendency of our technique to generate the clusters containing too few traces (too few traces means a very simple model), a minimum size θ of each potential cluster is requested to be set before starting the algorithm. Steps $4-9$ in Algorithm 1 check the number of traces in the original trace set and if there is no way to divide the trace set so that the sizes of both the subsets

Algorithm 1. Discovering the best solution for clustering traces (DBSCT)

Input: a set of traces T_C from event log C, the set of closed sequential patterns $CSP \leftarrow \Gamma(T_C, min\_sup)$ mined from T_C with a minimum threshold min_sup, the fitness weight α and complexity weight β for SMI, the minimum threshold μ for STB, the minimum thresholds μ_f and μ_c for SSTB, the minimum threshold φ_f for FCSTB, the maximum threshold φ_c for CCSTB, the minimum size θ for each cluster.

Let N, N_{left} and N_{right} be the nodes for a binary tree.

Let TB be a set of trace behaviors.

Let l be an array and $length(l) = 4$.

Let Φ be an algorithm which searches for the best *trace behavior* for splitting the original set of traces, the details about Φ is shown in Algorithm 2.

1: $N \leftarrow Null$ # create a node N
2: $N_{left} \leftarrow Null$ # N_{left} is the left child node of N
3: $N_{right} \leftarrow Null$ # N_{right} is the right child node of N
4: $l \leftarrow Null$
5: **if** $|T_C| \geq 2\theta$ **then**
6: $TB = TB \cup \Pi(CSP, T_C, \theta)$
7: **else**
8: return $N \leftarrow (T_C, \Omega(T_C), \Sigma_f(\Omega(T_C)), \Sigma_c(\Omega(T_C))$
label node N with a set of traces T, a process model $\Omega(T)$ and the quality information $\Sigma_f(\Omega(T))$ and $\Sigma_c(\Omega(T))$
9: **end if**
10: $l \leftarrow \Phi(TB, T_C, \alpha, \beta, \mu, \mu_f, \mu_c, \varphi_f, \varphi_c)$
11: **if** $l[trace\_behavior] = Null$ **then**
12: return $N \leftarrow (T_C, \Omega(T_C), \Sigma_f(\Omega(T_C)), \Sigma_c(\Omega(T_C)))$
13: **else**
14: $N_{left} \leftarrow DBSCT(l[trace\_set_1], CSP, \alpha, \beta, \mu, \mu_f, \mu_c, \varphi_f, \varphi_c, \theta)$
15: $N_{right} \leftarrow DBSCT(l[trace\_set_2], CSP, \alpha, \beta, \mu, \mu_f, \mu_c, \varphi_f, \varphi_c, \theta)$
16: **end if**
17: return $N \leftarrow (T_C, \Omega(T_C), \Sigma_f(\Omega(T_C)), \Sigma_c(\Omega(T_C)))$

Output: a binary tree bt with a root node N

generated are larger than or equal to θ then the algorithm stops. Afterwards, the trace behaviors discovered in step 6 are filtered and all the trace behaviors that can't lead to a valid division of the original trace set according to the minimum size rule are removed. Step 10 searches for the best trace behavior among all of the behaviors found in step 6 through the algorithm Φ depicted in Algorithm 2. A best trace behavior is defined as a behavior (either a SSTB or a FCSTB or a CCSTB) which can help generate a maximum sub-model improvement *SMI* as shown in the steps 12−13 in Algorithm 2. The main reason to set the parameter *SMI* is: if the average quality of the sub-models can't be improved to a certain extent based on the division procedure compared with the quality of the original model, then it is not worth making the division (this requirement stems from the consideration for the balance between the integrity and the quality of the process model). Algorithm 1 takes a greedy strategy for clustering the traces step by step, the same procedure continues on the subsets of traces generated by

Algorithm 2. Searching for the best trace behavior (Φ)

Input: a set of trace behaviors TB, a set of traces T, the fitness weight α and complexity weight β for SMI, the minimum threshold μ for STB, the minimum thresholds μ_f and μ_c for SSTB, the minimum threshold φ_f for FCSTB, the maximum threshold φ_c for CCSTB.
Let T_1, T_2 be two sets of traces.
Let p be an array and $length(p) = 4$.
Let $SSTB\_FCSTB\_CCSTB$ be an algorithm which judges if a trace behavior tb is either a SSTB or a FCSTB or a CCSTB.

1: T_1, $T_2 \leftarrow \emptyset$
2: $p[trace\_behavior] \leftarrow Null$; $p[smi] \leftarrow -\infty$; $p[trace\_set_1]$, $p[trace\_set_2] \leftarrow \emptyset$
3: **for** each trace behavior $tb \in TB$ **do**
4: **for** each trace $t \in T$ **do**
5: **if** $tb \sqsubseteq t$ **then**
6: $T_1 = T_1 \cup \{t\}$ # tb is a subsequence of t
7: **else**
8: $T_2 = T_2 \cup \{t\}$ # tb is not a subsequence of t
9: **end if**
10: **end for**
11: **if** $SSTB\_FCSTB\_CCSTB(T_1, T_2, T, \alpha, \beta, \mu, \mu_f, \mu_c, \varphi_f, \varphi_c)$ **then**
12: **if** $p[smi] \leq SMI(T_1, T_2, T, \alpha, \beta)$ **then**
13: $p[smi] \leftarrow SMI(T_1, T_2, T, \alpha, \beta)$
14: $p[trace\_set_1] \leftarrow T_1$
15: $p[trace\_set_2] \leftarrow T_2$
16: $p[trace\_behavior] \leftarrow tb$
17: **end if**
18: **end if**
19: **end for**
20: return p

Output: an array p which contains the information about the found trace behavior tb

the present stage as shown in the steps 11−15 in Algorithm 1. Finally, a binary tree bt is output by Algorithm 1 where each leaf node in bt represents a found cluster of traces.

3.4 Assumptions

In this paper we assume that the inaccurate and complex business process subjected to our method is able to be divided into several simpler and more accurate sub-processes where each sub-process carries out some specific functions. These functions are identified by certain behaviors of traces recorded in the event log.

4 Case Study

We tested the effectiveness of our technique on a real-life event log of the loan and overdraft approvals process from Business Process Intelligence Challenge 2012

(BPIC 2012). This log contains 13087 traces and 36 event classes. A process model (as shown in Figure 3) which has an ICS fitness equal to 0.9268 and a *Place/Transition Connection Degree* (PT-CD) equal to 3.939 is generated by using the *Heuristics Miner* on this log.

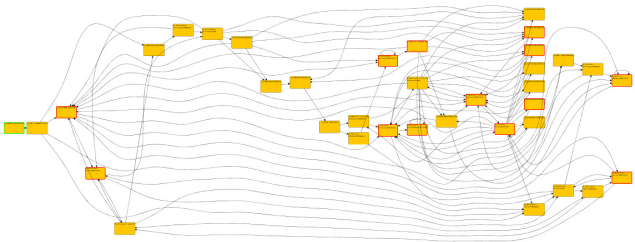


Fig. 3. Business process model of the loan and overdraft approvals process

Though this mined model has a good fitness, the whole model looks very complicated because it has a high complexity. So we set the fitness weight $\alpha = 0.4$ and the complexity weight $\beta = 0.6$ for calculating the *SMI*. The minimum support *min_sup* for the closed sequential pattern mining algorithm is set to 0.3, the minimum threshold μ for *SMI* is set to 0.04, both the minimum thresholds μ_f and μ_c for *SSTB* are set to 0.02, the minimum threshold φ_f for *FCSTB* is set to 0.84, the maximum threshold φ_c is set to 2.5 and the minimum size θ for each cluster is set to 655 (5% of the total number of traces in the event log). With these parameters set above, five clusters are generated by our technique. The weighted average ICS fitness[4], the weighted average control flows, the weighted average PT-CD and the weighted average number of and/xor join/splits for the process models mined from the traces clustered are calculated and shown in Table 1. The value on the right of the backslash in Table 1 is for the model mined by employing the original event log. Through the evaluation results exhibited in Table 1, we can see that the weighted average fitness of the sub-models from our technique is higher than the fitness of the original model, in the meantime the average complexity of these models has been greatly reduced. Such a result benefits from the thought of a trade-off between the accuracy and complexity in our technique.

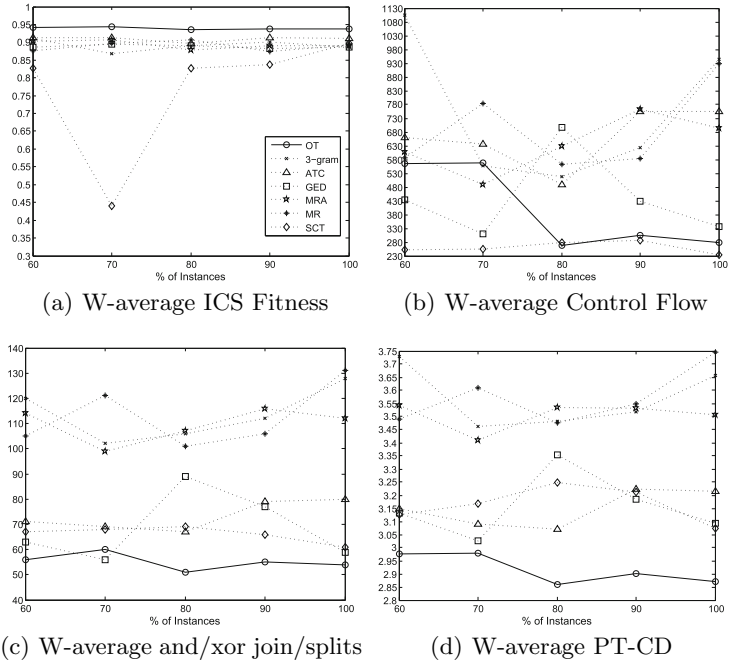
We also compared our technique (OT) to other six trace clustering techniques which are 3-gram [6], MR and MRA [5], ATC [3], GED [4] and sequence clustering (SCT) [7]. Except for the original event log, four random sub-logs that contain 60%, 70%, 80% and 90% of the instances from the original log have been chosen for analysis. The number of clusters for the six trace clustering techniques

[4] Let j be the number of clusters, m_i denotes the number of traces in cluster i , where $1 \leq i \leq j$. Let $ICS-F_i$ represents the ICS fitness of the process model for cluster i, the weighted average ICS fitness is defined as: $WAICS-F = \frac{\sum_{i=1}^{j} m_i \times ICS\text{–}F_i}{\sum_{i=1}^{j} m_i}$.

Table 1. Evaluation results for the sub-models from the traces clustered by using our technique

Weighted average ICS fitness	Weighted average control flows	Weighted average PT-CD	Weighted Average and/xor join/splits	Time (Min)
0.9371/0.9268(raw model)	279/1635	2.87/3.94	55/147	6.8

is set to equal to the number of clusters discovered by our technique. Figure 4 shows the results for the comparison.



(a) W-average ICS Fitness (b) W-average Control Flow

(c) W-average and/xor join/splits (d) W-average PT-CD

Fig. 4. Comparison among different trace clustering techniques

For the entire accuracy of the sub-process models discovered, our technique and the ATC perform better than other trace clustering techniques. The main reason is that both our technique and the ATC try to optimise the accuracy of the potential model for each cluster during the run time. However the ATC doesn't consider the complexity of process models during the clustering procedure. As a result, the entire complexity of the sub-process models discovered by ATC is much higher. The models discovered by utilising our technique are less complicated than the models mined by other techniques. The sequence clustering technique performs better than our technique on the evaluation related to weighted average control flow. Nevertheless, the accuracy of the models mined

by the sequence clustering technique is not as good as the accuracy of models discovered by our technique.

5 Related Work

In the literature, different trace clustering approaches have been put forward to overcome the negative impacts from high variety of behaviors stored in event logs. The basic idea is to transform the traces into vectors, then a distance metric can be defined among traces.

In [6] the authors propose an approach for expressing the traces by profiles so that a suitable environment can be built for clustering the traces. Each profile is a set of items that describe the trace from a specific perspective. Five profiles are recommended in [6] which are *activity profile*, *transition profile*, *case attributes profile*, *event attributes profile* and *performance profile*. By converting the profiles defined into an aggregate vector the distance between any two traces can be measured. One advantage of this technique is that it provides a full range of metrics for clustering traces.

In [14] the authors point out that a fully complete model (with high fitness) discovered may support a high variety of behaviors that are not registered in event log, as a result some significant structural features may be concealed in the mined model. Such a problem can be dealt with by considering the metric *soundness* [2] which measures the percentage of behaviors of the mined model that are recorded in the log among all of the behaviors supported by the model. An efficient technique is proposed in [14] which divides the whole process into a set of distinct sub-processes based on a greedy strategy which makes sure the further division of a process will lead to another increasingly sound sub-process. This method can also help solve the problem of high complexity of the initial model.

Context-aware trace clustering techniques are proposed in [5] and [4]. In [5] the authors indicate that the feature sets based on sub-sequences of different lengths are context-aware for the vector space model and can reveal some set of common functions. Two traces that have a lot of conserved features in common should be gathered in the same cluster. In [4] the authors present an edit distance-based approach for distributing traces into clusters such that each cluster consists of traces with similar structural behaviors. The cost of edit operations is associated with the contexts of activities so that the calculated edit distance between traces is more accurate.

The sequence clustering technique based on first-order Markov chains is presented in [7]. This technique learns a potential first-order Markov model for each cluster through an expectation-maximization algorithm. A sequence is assigned to a cluster which is able to generate it with higher probability. The technique proposed in this paper also inherits the idea from sequence clustering, the difference is our technique represents each cluster with a set of separate sequences (significant trace behaviors).

In [3] a novel technique for trace clustering is presented which directly optimises the fitness of each cluster's underlying process model during the run time.

This method doesn't consider the vector space model for trace clustering, it simply discovers the suitable traces for each cluster so that the combined accuracy of the related models for these clusters is maximised. This method sufficiently resolves the gap between the clustering bias and the evaluation bias.

6 Conclusion

In this paper we proposed a new trace clustering technique which is able to search for an optimal solution for clustering the traces from an event log among all of the possible solutions. This technique considers both the accuracy and complexity of the potential model for each cluster during the clustering procedure. Through the results from the experiment we demonstrated the effectiveness of our technique by comparing it with other six classical trace clustering techniques.

However, the technique presented in this paper encounters challenges on performance while dealing with event logs generated by totally unstructured business processes because such processes contain a tremendous number of behaviors. Our next main research task will be to focus on filtering the trival trace behaviors in the event logs so that the performance of our technique can be improved. In the meantime, we will also validate our methods on some other real-life cases.

References

1. van der Aalst, W.M.P., van Dongen, B.F., Herbst, J., Maruster, L., Schimm, G., Weijters, A.J.M.M.: Workflow Mining: A Survey of Issues and Approaches. Data and Knowledge Engineering **47**(2), 237–267 (2003)
2. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer, Heidelberg (2011)
3. Weerdt, J.D., vanden Broucke, S., Vanthienen, J., Baesens, B.: Active Trace Clustering for Improved Process Discovery. IEEE Transactions on Knowledge and Data Engineering **25**(12), 2708–2720 (2013)
4. Bose, R.P.J.C., 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. 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)
6. Song, M., Günther, C.W., van der Aalst, W.M.P.: Trace clustering in process mining. In: Ardagna, D., Mecella, M., Yang, J. (eds.) Business Process Management Workshops. LNBIP, vol. 17, pp. 109–120. Springer, Heidelberg (2009)
7. 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)
8. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann (August 2000)

9. Shengnan, C., Han, J., David, P.: Parallel mining of closed sequential patterns. In: KDD 2005 Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, pp. 562–567. ACM, New York
10. Mendling, J., Strembeck, M.: Influence factors of understanding business process models. In: BIS, pp. 142–153 (2008)
11. van der Aalst, W.M.P.: The Application of Petri Nets to Workflow Management. The J. Circuits, Systems and Computers **8**(1), 21–66 (1998)
12. Weijters, A.J.M.M., van der Aalst, W.M.P., Alves de Medeiros, A.K.: Process Mining with the Heuristics Algorithm. TU Eindhoven, BETA Working Paper Series 166 (2006)
13. de Medeiros, A.A.: Genetic Process Mining. Ph.D. thesis, Eindhoven University of Technology (2006)
14. Greco, G., Guzzo, A., Pontieri, L.: Discovering Expressive Process Models by Clustering Log Traces. IEEE Transactions on Knowledge and Data Engineering **18**(8), 1010–1027 (2006)