A Graph and Trace Clustering-based Approach for Abstracting Mined Business Process Models

Yaguang Sun and Bernhard Bauer
Software Methodologies for Distributed Systems, University of Augsburg, Augsburg, Germany

Keywords: Business Process Model Abstraction, Business Process Mining, Workflow Discovery, Graph Clustering, Trace Clustering.

Abstract: Process model discovery is a significant research topic in the business process mining area. However, existing workflow discovery techniques run into a stone wall while dealing with event logs generated from highly flexible environments because the raw models mined from such logs often suffer from the problem of inaccuracy and high complexity. In this paper, we propose a new process model abstraction technique for solving this problem. The proposed technique is able to optimise the quality of the potential high level model (abstraction model) so that a high-quality abstraction model can be acquired and also considers the quality of the sub-models generated where each sub-model is employed to show the details of its relevant high level activity in the high level model.

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 (van der Aalst et al., 2003). The starting point of these techniques is usually an event log which is a set of cases. A case is an instance of a business process and has an attribute *trace* which is a set of ordered events (each event is an instance of a specific activity). In the event log both cases and events are uniquely marked by *case id* and *event id* respectively (van der Aalst, 2011).

As one of the most important learning tasks in business process mining area, the current process model discovery techniques encounter great challenges in the context of real-life event logs. Such logs that usually contain a tremendous number of trace behaviors (expressed by the activities and their precedence relations in the trace) stem from the business processes executed in highly flexible environments, e.g., healthcare, customer relationship management (CRM) and product development (Weerdt et al., 2013). As a result, "spaghetti-like" business process models are often generated while mining real-life event logs with existing workflow discovery techniques. Such models are often inaccurate (in the process mining area the fitness is utilised to express the accuracy of a mined model which measures the proportion of behaviors in the event log possible according to the model) and difficult to be comprehended because of their high complexity. Accordingly, two main pioneering approaches have been developed in the literature to solve this problem: *trace clustering* technique (Weerdt et al., 2013; Bose and van der Aalst, 2009; Bose and van der Aalst, 2010; Song et al., 2009; Ferreira et al., 2007) and *process model abstraction-based* technique (Bose and van der Aalst, 2009; Baier and Mendling, 2013; Conforti et al., 2014).

Trace clustering techniques divide the raw event log into several sub-logs where each sub-log contains the traces with similar behaviors and helps generate a more accurate and comprehensible sub-model. Generally, these techniques perform well for handling the logs with a moderate amount of trace behaviors. Nevertheless, the limitation of current trace clustering techniques will be revealed while dealing with event logs containing massive trace behaviors. For instance, the event log of a Dutch academic hospital from Business Process Intelligence Contest 2011 (BPIC 2011) contains 624 activities among which a large number of relations are exhibited (the average out-degree for each activity is 6.2564) and most of the classical trace clustering methods can not bring a significant improvement on the mining result for this hospital log (as shown in Section 4).

Process model abstraction-based approaches

Sun, Y. and Bauer, B.
A Graph and Trace Clustering-based Approach for Abstracting Mined Business Process Models.
In *Proceedings of the 18th International Conference on Enterprise Information Systems (ICEIS 2016) - Volume 1*, pages 63-74
ISBN: 978-989-758-187-8

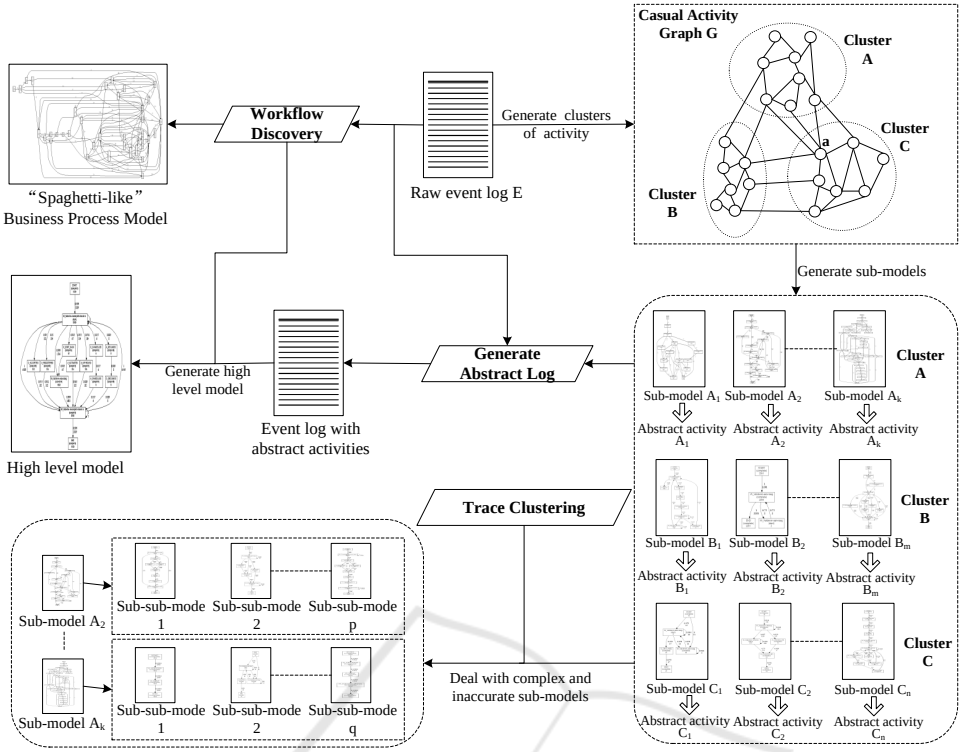

Figure 1: Illustration of the basic ideas of the approach proposed in this paper.

make the assumption that the raw models mined from real-life logs contain low level sub-processes which should be discovered in the form of sub-traces in the original event logs and abstracted into high level activities so that the insignificant low level process behaviors can be hidden in the high level activities. Thus, more accurate and simpler high level process models can be obtained. However, most of the present process model abstraction-based techniques focus mainly on the discovery of sub-processes and can not ensure the accuracy of the high level process models generated.

In this paper, we put forward a new method which inherits the characteristics of the trace clustering techniques and the process model abstraction-based approaches for solving the problem of "spaghetti-like" process models. The proposed technique is able to optimise the quality of the potential high level process model through a new abstraction strategy based on *graph clustering* technique (Schaeffer, 2007). As a result, a high-quality abstraction model can be built. Furthermore, the quality of the sub-models discovered for showing the details of their related high level activities (used for building the final high level model) is also considered by our approach. The structure of the main contents in this paper is organised as:

- A new strategy for abstracting the raw models mined from real-life event logs is discussed in Section 2.
- In Section 3.1, several important concepts that support the method proposed in this paper are reviewed. In Section 3.2, a three-stage model abstraction method based on the strategy proposed in Section 2 is elaborated.
- To test the efficiency of our method, we carry out a case study in Section 4 by applying our approach to three event logs: the repair log from (van der Aalst, 2011), the hospital log from Business Process Intelligence Contest (BPIC) 2011 and the log of the loan and overdraft approvals process from BPIC 2012.

2 BASIC IDEAS

In the real world, seemingly "spaghetti-like" business process models mined from event logs might still have some rules to follow. Sometimes, the main reason for the structurelessness of these mined models is that they contain several extremely complex substructures. However, the relations among these substructures may be straightforward (this is proven in the case study in Section 4). While turning to a specific event log, such kind of phenomenon mentioned

above can be reflected by the existence of several clusters of activities from an event log where the activities in the same cluster are densely connected and the activities in different clusters are sparsely connected (this is also the assumption for our method). For instance, in Figure 1 an event log E contains 22 activities and a *causal activity graph* G can be established by employing the activities from E as vertices and the *casual relations* (Hompes et al., 2014) among these activities as edges. The definitions about *casual activity graph* and *casual relations* of activities are introduced in detail in Section 3.1. According to Figure 1, the vertices in G can be grouped into three clusters by considering the edge structure in such a way that there should be many edges within each cluster and relatively few edges among the clusters.

With the assumption mentioned above, we put forward a new strategy for solving the problem of complex and inaccurate process models mined from real-life event logs. The basic idea is to generate the clusters of activities firstly by following the same rule utilised in the example shown in Figure 1. Afterwards, for each cluster one or several sub-models are generated where each sub-model only contains the activities from its relevant activity cluster. In the example from Figure 1, the sub-models for cluster A are built by using the activities from cluster A. Then, for a complex and inaccurate sub-model, trace clustering technique is employed to split it into several simple and accurate sub-sub-models so that the sub-model can be well comprehended. Finally, these sub-models (not including the sub-sub-models) generated are abstracted into high level activities with which a simple and accurate ultima high level process model is formed. In this paper the high level process model together with the sub-models (each sub-model is related to one high level activity in the high level model built) are used to show the details of the whole business process recorded in event log.

Basically, two major benefits could be acquired from the strategy proposed above. On one hand, the original tough problem (deal with the entire model) met by current trace clustering techniques is transformed into small sub-problems (deal with the sub-models). Specifically, the raw mined model from event log may contain too many behaviors which might be far beyond the abilities of existing trace clustering techniques. However, by distributing the huge amount of behaviors from the original mined model to several small sub-models (each sub-model contains less behaviors but still might be complex and inaccurate) the trace clustering techniques can provide better results while being applied on these sub-models. On the other hand, the number of activity relations among the clusters is kept as small as possible (which means the relations among the high level activities created are kept as few as possible). As a result, the quality of the potential high level process model is optimised to a large extent because it contains a limited number of behaviors among its activities.

3 APPROACH DESIGN

In this section, we propose a new approach that utilises the strategy introduced in Section 2 for solving the problem of "spaghetti-like" process models mined from event logs. In Section 3.1, several important basic concepts and notations related to our technique are discussed. In Section 3.2, the details of our technique are elaborated.

3.1 Preliminaries

Event logs (van der Aalst, 2011) play the significant part of data sources for various kinds of process mining techniques. The basic concepts related to event logs are conveyed by the following definitions.

Definition 1. *(Case)*
Let C be the set of cases. A case $c \in C$ is defined as a tuple $c = (N_c, \Theta_c)$, where $N_c = \{n_1, n_2, \ldots, n_k\}$ is the set of names of case attributes, $\Theta_c : N_c \rightarrow A_c$ is an attribute-transition function which maps the name of an attribute into the value of this attribute, where A_c is the set of attribute values for case c.

A case is an instance of a specific business process and uniquely identified by *case id*. Each case may have several attributes such as trace, originator, timestamp and cost, etc. As one of the most important case attributes, the *trace* of a case is defined as:

Definition 2. *(Trace)*
Let AT be the set of activities, EV be the set of events and each event $ev \in EV$ is an instance of a particular activity $at \in AT$. A trace is a sequence of ordered events from EV.

Definition 3. *(Event Log)*
An event log is defined as $E \subseteq C$, for any $c_1, c_2 \in E$ such that $c_1 \neq c_2$.

Take a simple event log $E_1 = [< a,b,c >^{15}, < a,c,b >^{15}, < a,b >^{3}, < a,c >^{5}]$ for example. This log contains 38 cases (only the case attribute *trace* is exhibited) and four kinds of trace[1]. There are totally

[1] A trace and a kind of trace are two different concepts. Each trace belongs to a unique case. A kind of trace contains several traces which have the same sequence of events.

$3 \cdot 15 + 3 \cdot 15 + 2 \cdot 3 + 2 \cdot 5 = 106$ events and three activities (activity a, b and c) in this log.

In (Hompes et al., 2014) the fundamental theory for *activity clustering* is developed. Two important concepts that support this theory are demonstrated: *Causal Activity Relations* and *Causal Activity Graph*. The technique proposed in this paper will use these concepts for generating the clusters of activities from event logs.

Definition 4. *(Direct and Casual Activity Relations)* Let AT be the set of activities of an event log E. Symbol $\succ_E$ represents a direct relation between two activities from AT and symbol $\succeq_E$ represents a causal relation between two activities from AT. Let $a, b \in AT$ be two activities, $\phi \in [-1.0, 1.0]$ be a threshold, $a \succ_E b = true$ if $|a \succ_E b| > 0$, where $|a \succ_E b|$ is the number of times that a is directly followed by b in E. $a \succeq_E b = true$ if $|a \succeq_E b| \geq \phi$, where $|a \succeq_E b| \in [-1.0, 1.0]$ is the value of casual relation between a and b.

In our approach we utilise the *DependencyMeasure* method introduced in (Weijters et al., 2006) for calculating the value of casual relation between any two activities which is defined as:

$$|a \succeq_E b| = \begin{cases} \frac{|a \succ_E b| - |b \succ_E a|}{|a \succ_E b| + |b \succ_E a| + 1} & \text{if } a \neq b \\ \frac{|a \succ_E a|}{|a \succ_E a| + 1} & \text{if } a = b \end{cases} \quad (1)$$

A $|a \succeq_E b|$ value close to 1.0 implies a high possibility that there exists a direct casual relation between a and b while a value close to –1.0 signifies a high possibility that there exists no casual relation between a and b. A value close to 0 means uncertainty. Take two activities a and c from event log E_1 created above as an example, $|a \succ_{E_1} c| = 15 + 5 = 20$, $|c \succ_{E_1} a| = 0$, so $|a \succeq_{E_1} c| = (20 - 0)/(20 + 0 + 1) \approx 0.95$. Let the threshold $\phi = 0.9$, then a casual relation is judged to exist between a and c because $|a \succeq_{E_1} c| > \phi$.

Definition 5. *(Casual Activity Graph)* Let AT be a set of activities from event log E, $\Upsilon(AT)$ denotes the set of casual activity graphs over AT. A causal activity graph $G \in \Upsilon(AT)$ is a tuple $G = (V, L)$ where $V \in AT$ is the set of vertices and $L \in (V \times V)$ is the set of edges. Each edge in G represents a casual relation between two activities.

In our method we employ an existing graph clustering technique (based on energy model) from (Noack, 2007) for mining the casual activity graphs following the rule that the activities in the same cluster should be densely connected and the activities in different clusters should be sparsely connected. The main reason for us to select this graph clustering technique is that it is able to automatically generate a suitable number of clusters of vertices according to the edge structure of a graph and also has a good performance. The basic knowledge related to graph clustering technique is well introduced in (Schaeffer, 2007).

3.2 A Three-step Algorithm

In this section a process model abstraction algorithm that consists of three main stages is put forward. This algorithm applies the strategy mentioned in Section 2 which considers the quality of both the potential high level model and sub-models generated. Let $\Pi : (SE, STH) \longrightarrow SG$ be a casual activity graph building method, where *SE* is the set of event logs, *STH* is the set of values of thresholds for judging casual relations among activities and *SG* is the set of casual activity graphs, $\Gamma : SG \longrightarrow SC$ be the graph clustering algorithm from (Noack, 2007), where *SC* is the set of all sets of activity clusters. The details of our method is described in Algorithm 1.

Algorithm 1: Abstracting the raw models mined (AM).

Input: an event log E, the threshold ϕ for judging the causal relations among activities, the threshold α for judging if a high level activity generated should be removed or not, the threshold β for searching for merging modes, a sub-model complexity threshold τ and a sub-model accuracy threshold χ, a trace number threshold κ, cluster number n.

Let G be a casual activity graph.
Let C_{ac} be a set of activity clusters.

1: $G \leftarrow Null$
2: $C_{ac} \leftarrow Null$
3: $G = \Pi(E, \phi)$ # build the casual activity graph
4: $C_{ac} = \Gamma(G)$ # mine the activity clusters
5: ***Stage 1:*** Find multi-cluster activities and extract sub-logs.
input: E, C_{ac}.
output: a new set of activity clusters *MC–*C_{ac}, a set of sub-logs *SSE*.
6: ***Stage 2:*** Generate high level activities and high level model.
input: *SSE*, E, α, β.
output: a high level model *HL–M*, a set of high level activities *H–SA*, a set of sub-logs *H–SSE*.
7: ***Stage 3:*** Deal with complex and inaccurate sub-models from *H–SSE*.
input: *H–SSE*, τ, χ, κ, n.
output: a set of sub-models *SSM*.

Output: a high level model *HL–M*, a set of sub-models *SSM*.

3.2.1 Find Multi-cluster Activities and Extract Sub-logs

In this subsection we make the assumption that a set of activity clusters $C_{ac} = \{c_1, c_2, \ldots, c_m\}$ for event log E has been acquired by Algorithm 1. Sometimes, an activity $a \in c_k \in C_{ac}$ may also have a lot of casual relations with the activities from other clusters. For instance, in the casual activity graph G from Figure 1, the activity a that pertains to cluster C is also connected to many activities in cluster A. In the graph clustering research area most of the classical methods developed presume that a vertice of a graph only belongs to one specific cluster. The graph clustering algorithm utilised in our approach also has the same assumption. However, it is a normal situation that some activities in a casual activity graph should pertain to more than one clusters according to the edge structure of the graph. Based on this fact, we develop a new concept named *Multi-cluster Activity* which is defined as:

Definition 6. *(Multi-cluster Activity)*
Let $\Phi : SG \longrightarrow SV$ be a graph density calculation schema, where SG is the set of casual activity graphs and SV is the set of values of graph density. Given a set of activity clusters $C_{ac} = \{c_1, c_2, \ldots, c_n\}$, an activity $a \in c_k \in C_{ac}$ is a multi-cluster activity if $\exists c_m \in C_{ac}$ such that $\Phi(G_m^{'}) \geq \Phi(G_m)$, where $G_m = (V_m, L_m)$ represents the casual activity graph built by using the activities from activity cluster c_m and $G_m^{'} = (V_m \cup a, L_m^{'})$ is a new graph generated by adding the activity a in G_m.

Given a graph $G = (V,L)$, $\Phi(G) = |L|/(|V| \times (|V|-1))$, where $|L|$ and $|V|$ stand for the total number of edges and the total number of vertices in graph G respectively. The main reason to use graph density for judging a multi-cluster activity is that densely connected activities are more likely to cause complex process behaviors that can't be expressed by the utilised workflow discovery algorithms (our approach leave these potential complex behaviors to trace clustering techniques). Our method detects all of the multi-cluster activities in C_{ac} and then distributes each of them to the eligible activity clusters in C_{ac} so that a new set of activity clusters $MC{-}C_{ac}$ can be generated. For example, let $C_{ac}^{'} = \{c_1, c_2, c_3\}$ be a set of activity clusters mined from event log $E^{'}$, $c_1 = \{a,b,c\}$, $c_2 = \{d,e\}$ and $c_3 = \{f,g,h\}$, pretend that $\Phi(G_{c_2}) = 0.5$, $\Phi(G_{c_3}) = 0.8$, $\Phi(G_{c_2}^{+}) = 0.63$ and $\Phi(G_{c_3}^{+}) = 0.7$, where G_{c_2} is the casual graph for cluster c_2, G_{c_3} for cluster c_3, $G_{c_2}^{+}$ is the casual graph generated by adding the activity $a \in c_1$ in G_{c_2} and $G_{c_3}^{+}$ generated by adding the activity a in G_{c_3}. According to Definition 6, a is a multi-cluster activity because $\Phi(G_{c_2}^{+}) > \Phi(G_{c_2})$. Afterwards, a new activity cluster $c_2^{'} = \{a,d,e\}$ is generated by adding a in c_2. Activity a should not be added in c_3 because $\Phi(G_{c_3}^{+}) < \Phi(G_{c_3})$. Let's presume that a is the only multi-cluster activity found, then the new set of activity clusters $MC{-}C_{ac}^{'} = \{c_1, c_2^{'}, c_3\}$ can be generated.

An intuitive proof about the benefit for locating the multi-cluster activities is shown in the example in Figure 1. We assume that the activity a in cluster C is a multi-cluster activity corresponding to cluster A. By adding a to cluster A the original casual graph G can be transformed into $G^{'}$ as shown in Figure 2. In $G^{'}$, the interrelations between cluster A and C are further decomposed which helps improve the quality of the potential high level model.

Whereafter, the stage 1 of Algorithm 1 creates a sub-log for each activity cluster in $MC-C_{ac} = \{mc_1, mc_2, \ldots, mc_n\}$. For example, for the activity cluster $mc_k \in MC-C_{ac}$ a new log E_{mc_k} is built which contains all of the sub-traces extracted from the original event log E where each sub-trace only includes the activities from mc_k. For instance, let $MC{-}C_{ac}^{'} = \{\{a,b,v,c,d\}, \{u,v,x,z\}\}$ be a set of activity clusters generated by stage 1 of Algorithm 1 executed on an event log $E^{'} = \{<a,b,c,d,v,x,z>^{80}, <a,c,d,u,v,x,z>^{150}, <a,b,v,c,d,u,v,z>^{200}\}$ (pretend that v is a multi-cluster activity). For the activity cluster $\{a,b,v,c,d\} \in MC{-}C_{ac}^{'}$ a new sub-log $SE_1^{'} = \{<a,b,c,d,v>^{80}, <a,c,d>^{150}, <a,b,v,c,d>^{200}\}$ can be created by extracting all the sub-traces in $E^{'}$ where these sub-traces only contain the activities from $\{a,b,v,c,d\}$. Similarly, the sub-log $SE_2^{'} = \{<v,x,z>^{80}, <u,v,x,z>^{150}, <u,v,z>^{200}\}$ can be generated for activity cluster $\{u,v,x,z\}$.

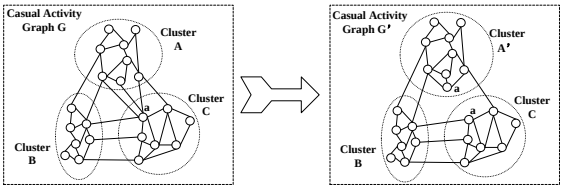

Figure 2: Further decompose the interrelations between cluster A and cluster C.

3.2.2 Generate High Level Activities and High Level Process Model

We presume that the set of sub-logs $SSE = \{E_{mc_1}, E_{mc_2}, \ldots, E_{mc_n}\}$ has been output by the stage 1 of Algorithm 1. Let $\Psi : SE \longrightarrow SS{-}SE$ be a method which splits an event log into several sub-logs where each sub-log contains the traces with the same start

activity and end activity, *SE* represents the set of event logs and *SS−SE* represents the set of all set of sub-logs. Take the simple event log $E^{'} = \{< a,b,c >^{15}, < a,d,c >^{15}, < a,f >^{3}, < a,e,d >^{5}\}$ as an example, $\Psi(E^{'}) = \{E^{'}_1, E^{'}_2, E^{'}_3\}$, where $E^{'}_1 = \{< a,b,c >^{15}, < a,d,c >^{15}\}$, $E^{'}_2 = \{< a,f >^{3}\}$ and $E^{'}_3 = \{< a,e,d >^{5}\}$. The high level activity generation method for the stage 2 of Algorithm 1 is depicted in Algorithm 2.

Algorithm 2: Generate high level activities (GHLA).

Input: the set of sub-logs *SSE*, the original log E, a threshold α, a threshold β.
Let Λ be a trace merging technique which is described in Algorithm 3.
Let *H−SSE* be a set of sub-logs where each sub-log $HE_q \in H{-}SSE$ is relevant to one potential high level activity.
Let *H−SA* be a set of high level activities.
Let *M−SSE* be a set of event logs with merged traces.

1: $H{-}SSE \leftarrow Null$
2: $H{-}SA \leftarrow Null$
3: $M{-}SSE \leftarrow Null$
4: **for** each log $E_{mc_k} \in SSE$ **do**
5: $M{-}SSE \leftarrow M{-}SSE \cup \Lambda(E_{mc_k}, SSE, E, \beta)$
generate log with merged traces.
6: **end for**
7: **for** each log $ME_p \in M{-}SSE$ **do**
8: $H{-}SSE \leftarrow H{-}SSE \cup \Psi(ME_p)$
9: **end for**
10: **for** each log $HE_q \in H{-}SSE$ **do**
11: $H{-}SA \leftarrow H{-}SA \cup HL{-}Activity(q)$
create a high level activity called
$HL{-}Activity(q)$ and put it in $H{-}SA$.
12: **end for**
13: **for** each $HL{-}Activity(p) \in H{-}SA$ **do**
14: **if** $|HL{-}Activity(p)| < \alpha$ **then**
15: remove $HL{-}Activity(p)$ from $H{-}SA$
16: remove HE_p from $H{-}SSE$
$|HL{-}Activity(p)|$ represents the
frequency of occurrence for the
high level activity $HL{-}Activity(p)$.
17: **end if**
18: **end for**

Output: the set of high level activities *H−SA*, the set of sub-logs *H−SSE*.

To explain Algorithm 2 explicitly, an example is employed here (for the rest part of this sub-section). Let $MC{-}C^{'}_{ac} = \{\{a,b,c,d\},\{u,v,x,z\}\}$ be a set of activity clusters generated by stage 1 of Algorithm 1 executed on an event log $E^{'} = \{< a,b,d,u,x,z >^{100}, < a,b,c,d,v,x,z >^{80}, < a,c,d,u,v,x,z >^{150}, < a,b,v,c,d,u,x,z >^{8}\}$, $SSE^{'} = \{E^{'}_{mc_1}, E^{'}_{mc_2}\}$ be a set of sub-logs generated by stage 1 of Algorithm 1 with inputs $MC{-}C^{'}_{ac}$ and $E^{'}$, where sub-log $E^{'}_{mc_1} = \{< a,b,d >^{100}, < a,b,c,d >^{80}, < a,c,d >^{150}, < a,b >^{8}, < c,d >^{8}\}$, sub-log $E^{'}_{mc_2} = \{< u,x,z >^{108}, < v,x,z >^{80}, < u,v,x,z >^{150}, < v >^{8}\}$. A set of sub-logs $H{-}SSE^{'} = \{\{< a,b,d >^{100}, < a,b,c,d >^{80}, < a,c,d >^{150}\}_0, \{< a,b >^{8}\}_1, \{< c,d >^{8}\}_2, \{< u,x,z >^{108}, < u,v,x,z >^{150}\}_3, \{< v,x,z >^{80}\}_4, \{< v >^{8}\}_5\}$ can be generated if $SSE^{'}$ is directly dealt with by the steps $7-9$ of Algorithm 2 (replace the set $M{-}SSE$ in step 7 by using $SSE^{'}$). Afterwards, according to the steps $10-12$ of Algorithm 2 a set of high level activities $H{-}SA^{'} = \{HL{-}Activity(0)^{330}, HL{-}Activity(1)^{8}, HL{-}Activity(2)^{8}, HL-Activity(3)^{258}, HL-Activity(4)^{80}, HL-Activity(5)^{8}\}$ is generated where each high level activity is related to a specific sub-log in $H{-}SSE^{'}$. In our method a high level activity will replace all the sub-traces that exist in its relevant sub-log in $H{-}SSE^{'}$ in the original event log $E^{'}$. For instance, the high level activity $HL{-}Activity(0)$ will replace all the sub-traces from the sub-log $\{< a,b,d >^{100}, < a,b,c,d >^{80}, < a,c,d >^{150}\}_0$ in $E^{'}$. Finally, a high level event log $E^{'}_h = \{< HL-Activity(0), HL-Activity(3) >^{100}, < HL-Activity(0), HL-Activity(4) >^{80}, < HL-Activity(0), HL-Activity(3) >^{150}, < HL-Activity(1), HL-Activity(5), HL-Activity(2), HL-Activity(3) >^{8}\}$ is acquired. The steps $13-18$ of Algorithm 2 remove all the infrequent high level activities generated and their relevant sub-logs in $H{-}SSE^{'}$ either. Removing infrequent activities which is in accordance with the main idea of most advanced process model mining techniques can make the potential model mined concentrate on exhibiting the most frequent process behaviors. In our example, given a threshold $\alpha = 20$, the high level activity $HL{-}Activity(1)$, $HL{-}Activity(2)$ and $HL{-}Activity(5)$ are removed from $H{-}SA^{'}$ and $E^{'}_h$ because the value of their frequency is eight which is smaller than α. At the same time, the sub-logs $\{< a,b >^{8}\}$, $\{< c,d >^{8}\}$ and $\{< v >^{8}\}$ are removed from $H{-}SSE^{'}$. Afterwards, a high level model can be built by mining the generated high level event log $E^{'}_h$ with an existing process model discovery algorithm (this is the way for our method to generate a high level model). Each sub-log in $H{-}SSE^{'}$ will be used to build a sub-model for indicating the details of its relevant high level activity.

Such a design for generating the high level activities will help maintain the precision (van der Aalst,

2011) (precision quantifies the ratio of the behaviors that can be generated by the mined models which are also recorded in the event logs) of the potential high level model together with the sub-models generated compared with the precision of the model mined by using the original log $E^{'}$ (the interested reader can think about it more deeply). Furthermore, our method might generate a huge amount of high level activities while encountering the event logs that have casual graphs with uniform structures. So we make the assumption that the casual graphs of the event logs processed by our method have structures with natural clusters.

Three infrequent high level activities ($HL-Activity(1)$, $HL-Activity(2)$ and $HL-Activity(5)$) are generated in the example mentioned above. This is because activity v happens between activity b and c in some traces in $E^{'}$ infrequently and v belongs to a different activity cluster from b and c. As a result, three kinds of infrequent sub-trace $<a,b>$, $<v>$ and $<c,d>$ in $H-SSE^{'}$ are generated by our method. The Algorithm 2 will remove all infrequent high level activities and also the sub-logs related to these activities. A lot more activities like activity v might lead to the situation that a huge amount of process behaviors in the original event logs will get lost because of being distributed into many infrequent sub-logs in $H-SSE$ which then will be removed. In this paper we propose a trace merging approach (called Λ which appears in the step 5 of Algorithm 2 and helps preserve the process behaviors recorded in the original logs as many as possible) for fixing this problem by employing the following definitions:

Definition 7. *(merging mode)*
Let $SSE = \{E_{mc_1}, E_{mc_2}, \ldots, E_{mc_n}\}$ be a set of sub-logs output by stage 1 of Algorithm 1 executed on an event log E. Let st_1 and st_2 be two sub-traces from $E_{mc_k} \in SSE$, sa_1 be the starting activity of st_1 and ea_2 be the ending activity of st_2. The pair (st_1, st_2) is called a merging mode for E_{mc_k} if (1) $|st_1| < \beta \times |E_{mc_k}|$ and $|st_2| < \beta \times |E_{mc_k}|$ where $|st_1|$ represents the total number of traces in E_{mc_k} which have the same event sequence as st_1, $|st_2|$ represents the total number of traces which have the same event sequence as st_2 and $|E_{mc_k}|$ represents the total number of traces in E_{mc_k}, (2) st_1 and st_2 appear in the same trace from E in the way $<st_1, \ldots, st_2>$, (3) the number of traces in E_{mc_k} which have sa_1 as starting activity and ea_2 as ending activity at the same time is larger than or equal to $\beta \times |E_{mc_k}|$.

Definition 8. *(minimum merging mode)*
Let (st_1, st_2) be a merging mode for a sub-log $E_{mc_k} \in SSE$, sa_1 be the starting activity of st_1 and ea_2 be the ending activity of st_2, $<st_1, \ldots, st_2>$ be a sub-trace from the original log E. The merging mode (st_1, st_2) is called a minimum merging mode if there exists no other merging modes in the sub-trace $<st_1, \ldots |st_2>$ or in the sub-trace $<st_1 | \ldots, st_2>$, where $<st_1, \ldots |st_2>$ represents a sub-trace generated by removing st_2 from $<st_1, \ldots, st_2>$ and $<st_1 | \ldots, st_2>$ by removing st_1 from $<st_1, \ldots, st_2>$.

For the example mentioned above, given a threshold $\beta = 0.05$, the pair $(<a,b>, <c,d>)$ from $E^{'}_{mc_1}$ is a merging mode (there are eight of such merging modes) because there are 330 traces in $E^{'}_{mc_1}$ that have activity a as starting activity and activity d as ending activity which is larger than $\beta \times |E^{'}_{mc_1}| = 17.3$. In the meantime, $|<a,b>| = 8 < 17.3$ and $|<c,b>| = 8 < 17.3$. Furthermore, the way for the sub-traces $<a,b>$ and $<c,d>$ to appear in the trace $<a,b,v,c,d,u,x,z>$ from $E^{'}$ also satisfies the condition proposed in Definition 7. The merging mode $(<a,b>, <c,d>)$ is also a minimum merging mode according to Definition 8.

Algorithm 3: Merging Traces (Λ).

Input: the set of sub-logs SSE, a sub-log $E_{mc_k} \in SSE$, a threshold β.
Let SMD be a set of merging modes.
1: $SMD \leftarrow Null$
2: **for** each sub-trace $st_p \in E_{mc_k}$ **do**
3: **if** st_p doesn't pertain to any merging mode in SMD **then**
4: **if** there exists another sub-trace $st_q \in E_{mc_k}$ and (st_p, st_q) is a merging mode **then**
5: put (st_p, st_q) in SMD
6: put the related sub-trace $<st_p, \ldots, st_q>$ from E in E_{mc_k}
7: remove st_p and st_q from E_{mc_k}
8: remove the sub-traces that appear between st_p and st_q in $<st_p, \ldots, st_q>$ from their original places in SSE
9: **end if**
10: **else**
11: *continue*
12: **end if**
13: **end for**
Output: the sub-log E_{mc_k} with merged traces.

With the two definitions created above, the details of the trace merging technique Λ is described in Algorithm 3. Here we still use the last example to explain how Λ works. As is shown that three infrequent high level activities are generated by running the Algorithm 2 directly starting from step 7 in our example. One intuitive method to solve this problem is to find all minimum merging modes in $SSE^{'}$ and then merge the sub-traces in the same merging

mode (reflected by the steps $2-13$ of Algorithm 3 and the steps $4-9$ of Algorithm 2). For example, eight merging modes $(<a,b>,<c,d>)^8$ for E'_{mc_1} can be constituted (given a threshold $\beta = 0.05$) and each pair of the sub-traces should be merged into a single sub-trace $<a,b,v,c,d>$ (eight of such merged sub-traces can be generated). Then, a new set of sub-logs $M{-}SSE' = \{ME'_1, ME'_2\}$ can be formed, where $ME'_1 = \{<a,b,d>^{100}, <a,b,c,d>^{80}, <a,c,d>^{150}, <a,b,v,c,d>^8\}$ and $ME'_2 = \{<u,x,z>^{108}, <v,x,z>^{80}, <u,v,x,z>^{150}\}$ (ME'_2 doesn't contain the kind of sub-trace $<v>$ any more because all of them are merged into the kind of sub-trace $<a,b,v,c,d>$ in ME'_1). Afterwards, by using the steps $7-18$ of Algorithm 2 to deal with the $M{-}SSE'$ a new set of sub-logs $H{-}SSE' = \{\{<a,b,d>^{100}, <a,b,c,d>^{80}, <a,c,d>^{150}, <a,b,v,c,d>^8\}_0, \{<u,x,z>^{108}, <u,v,x,z>^{150}\}_1, \{<v,x,z>^{80}\}_2\}$ and a new set of high level activities $H{-}SA' = \{HL{-}Activity(0)^{338}, HL{-}Activity(1)^{258}, HL{-}Activity(2)^{80}\}$ can be generated. Now no infrequent high level activities exist in $H{-}SA'$ any longer.

3.2.3 Deal With Complex and Inaccurate Sub-models

In this subsection we presume that a set of sub-logs $H{-}SSE$ has been output by the stage 2 of Algorithm 1. For each sub-log in $H{-}SSE$ a sub-model is mined with existing workflow discovery technique to depict the details of the sub-log's relevant high level activity. In our approach, the business process recorded in an event log is expressed by the generated high level model and the sub-models together. However, the strategy (mentioned in Section 2) used in our method try to decrease the number of behaviors in the potential high level model by hiding most of the original process behaviors inside the high level activities generated. As a result, the sub-models for the high level activities might still be complex and inaccurate. Trace clustering technique is utilised for solving this problem.

Let $\Omega : S{-}E \rightarrow S{-}M$ be a workflow discovery algorithm, where $S{-}M$ is the set of process models and $S{-}E$ is the set of event logs, $\Theta_{accuracy} : (S{-}E, S{-}M) \rightarrow SV_{accuracy}$ be a process model accuracy evaluation method, where $SV_{accuracy}$ is the set of accuracy values of the mined process models, $\Theta_{complexity} : S{-}M \rightarrow SV_{complexity}$ be a process model complexity evaluation method, where $SV_{complexity}$ is the set of complexity values of the mined process models. Let $T_{clustering} : (S{-}E, SV_{cnumber}) \rightarrow SS{-}E$ be a trace clustering algorithm, $SS{-}E$ is the set of all sets of sub-logs and $SV_{cnumber}$ is the set of numbers of the clusters generated. The main procedure for dealing with the low-quality sub-models mined is depicted in Algorithm 4.

Algorithm 4: Deal with low-quality sub-models.

Input: the set of sub-logs $H{-}SSE$, a sub-model complexity threshold τ and a sub-model accuracy threshold χ, a trace number threshold κ, cluster number n.

Let SSM, SSM_c be two sets of sub-models.
Let $S{-}E_c$ be a set of sub-logs.
Let m_1, m_2 be two variants of float type.
Let m_3 be a variant of int type.

1: $SSM \leftarrow Null$, $SSM_c \leftarrow Null$
2: $S{-}E_c \leftarrow Null$
3: $m_1 \leftarrow 0$, $m_2 \leftarrow 0$
4: $m_3 \leftarrow 0$
5: **for** each sub-log $SE \in H{-}SSE$ **do**
6: **if** $\Theta_{accuracy}(\Omega(SE), SE) < \chi$ $||$ $\Theta_{complexity}(\Omega(SE), SE) > \tau$ && $|SE| \geq \kappa$ **then**
7: $S{-}E_c = T_{clustering}(SE, n)$
8: **for** each sub-log $SE_c \in S{-}E_c$ **do**
9: $SSM_c \leftarrow SSM_c \cup \Omega(SE_c)$
10: $m_1 \leftarrow m_1 + \Theta_{accuracy}(\Omega(SE_c), SE_c)|SE_c|$
11: $m_3 \leftarrow m_3 + |SE_c|$
12: **end for**
13: $m_2 \leftarrow m_1/m_3$ # calculate weighted average # accuracy
14: **if** $m_2 \geq \Theta_{accuracy}(\Omega(SE), SE)$ **then**
15: **for** each sub-model $SM_c \in SSM_c$ **do**
16: $SSM \leftarrow SSM \cup SM_c$
17: **end for**
18: **else**
19: $SSM \leftarrow SSM \cup \Omega(SE)$
20: **end if**
21: $m_1 \leftarrow 0$, $m_3 \leftarrow 0$, $SSM_c \leftarrow Null$
22: **else**
23: $SSM \leftarrow SSM \cup \Omega(SE)$
24: **end if**
25: **end for**

Output: the set of sub-models SSM.

According to Algorithm 4, a sub-log SE from $H{-}SSE$ that leads to a low-quality sub-model M (the quality is judged by using the sub-model accuracy and complexity thresholds χ and τ in the step 6 of Algorithm 4) will be divided into n sub-sub-logs by using the trace clustering technique if the number of the traces inside SE is larger than or equal to a threshold κ. Afterwards, for each sub-sub-log a sub-sub-model is built (in the step 9 of Algorithm 4). If the weighted average accuracy of the sub-sub-models generated is larger than or equal to the accuracy of the original

sub-model then these sub-sub-models are added to the set of sub-models *SSM* which will be finally output by Algorithm 4 (Algorithm 4 will not use the sub-sub-models if their weighed average accuracy is lower than the accuracy of their related original sub-model). If a sub-log $SE^{'}$ from $H{-}SSE$ leads to a good-quality sub-model $M^{'}$ then add $M^{'}$ in *SSM* (step 23 of Algorithm 4).

The authors in (Weerdt et al., 2013) develop a metric called *Place/Transition Connection Degree* (PT-CD) for quantifying the complexity of a Petri net which is defined as:

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

In Equation 2, $|a|$ represents the total number of arcs in the process model, $|P|$ is the number of places and $|T|$ is the number of transitions. The greater the PT-CD is, the more complicated the model will be.

In this paper, we utilise the *Heuristics Miner* (HM) (Weijters et al., 2006) for generating the process models. The ICS fitness developed in (de Medeiros, 2006) is utilised for evaluating the accuracy of the mined heuristic net. Then, the *Heuristic Net to Petri Net* plugin in ProM 6[2] is used for transforming the heuristic net mined into a Petri net. Afterwards, the *PT-CD* is employed for evaluating the complexity of the Petri net obtained. The trace clustering technique *GED* from (Bose and van der Aalst, 2009) is utilised for dividing the sub-logs in $H{-}SSE$ into sub-sub-logs (step 7 of Algorithm 4).

4 CASE STUDY

We tested the effectiveness of our approach on three event logs: the repair log (Repair) from (van der Aalst, 2011), the hospital log (Hospital) from BPIC 2011 (in our experiment an artificial start activity and end activity are added in the traces from the hospital log) and the log of the loan and overdraft approvals process (Loan) from BPIC 2012. The basic information about the three logs is shown in Table 1. The quality information of the models mined from the three logs by using HM is listed in Table 2. Except for *Place/Transition Connection Degree* (PT-CD) mentioned in the last section, another process model complexity metric is also used for evaluating the complexity of the mined models in our experiment which is *Extended Cardoso Metric* (E-Cardoso) (Lassen and van der Aalst, 2009).

Firstly, six classical trace clustering techniques are executed on the three logs which are 3-gram (Song et al., 2009), MR and MRA (Bose and van der Aalst, 2010), ATC (Weerdt et al., 2013), GED (Bose and van der Aalst, 2009) and sequence clustering (SC) (Ferreira et al., 2007). For each trace clustering approach six sub-logs are generated for every of the three logs utilised. The assessment results on these techniques are shown in Table 3. The metric $W_t - ICS$ stands for the weighted average ICS fitness based on the number of traces and $W_e - ICS$ represents the weighted average ICS fitness based on the number of events. For example, let $S{-}E = \{E_1, E_2, E_3, E_4, E_5, E_6\}$ be a set of sub-logs output by a trace clustering technique carried out on event log E. For a sub-log $E_k \in S{-}E$, $|E_k|_t$ represents the total number of traces in E_k, $|E_k|_e$ represents the total number of events in E_k and ICS_{E_k} represents the value of ICS fitness for the sub-model mined from sub-log E_k. Then, the $W_t - ICS$ for the sub-logs in $S{-}E$ is equal to $(\sum_{k=1}^{6} |E_k|_t \times ICS_{E_k}) / \sum_{k=1}^{6} |E_k|_t$ and the $W_e - ICS$ is equal to $(\sum_{k=1}^{6} |E_k|_e \times ICS_{E_k}) / \sum_{k=1}^{6} |E_k|_e$. According to the evaluation results shown in Table 3, most trace clustering techniques perform well on the log Repair which contains the least trace behaviors among the three logs. Nevertheless, for the logs Loan and Hospital which have more trace behaviors most trace clustering techniques employed could not bring a significant improvement on the accuracy of the mined models (especially for the log Hospital).

Whereafter, the approach proposed in this paper is evaluated by using the three logs mentioned above. The threshold ϕ for judging the casual relations is set to zero (such a setting will help find more complete activity clusters), the threshold α for judging whether a high level activity generated should be removed or not is set to 20, the threshold β for searching for the merging modes is set to 0.05, the sub-model complexity threshold τ (for PT-CD) is set to 2.5, the sub-model accuracy threshold χ (for ICS fitness) is set to 0.8, the trace number threshold κ is set to 100 and the number of clusters for the trace clustering technique *GED* is

Table 1: Basic information of the evaluated logs.

Log	Traces	Events	Event types
Repair	1000	10827	12
Loan	13087	262200	36
Hospital	1143	150291	624

Table 2: Evaluation results for the models mined by using the log Repair, Loan and Hospital.

Log	ICS	$E-Cardoso$	$PT-CD$
Repair	0.6768	31	2.3656
Loan	0.7878	148	3.1478
Hospital	0.6058	2108	2.703

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

Table 3: Evaluation results for the six classical trace clustering techniques executed on the log Repair, Loan and Hospital.

Log	Method	$W_t - ICS$	$W_e - ICS$
Repair	3-gram	0.9299	0.9326
	MR	0.8123	0.814
	MRA	0.8056	0.8055
	ATC	0.9971	0.996
	GED	0.7908	0.7907
	SC	0.9823	0.9802
Loan	3-gram	0.7965	0.7282
	MR	0.7828	0.6984
	MRA	0.8181	0.7285
	ATC	0.7653	0.5665
	GED	0.8038	0.7992
	SC	0.9255	0.9164
Hospital	3-gram	0.6153	0.69
	MR	0.5785	0.6622
	MRA	0.5629	0.6844
	ATC	0.7583	0.705
	GED	0.6003	0.6837
	SC	0.7354	0.7129

set to 6. The quality information of the sub-models generated is shown in Table 4, the quality information of the three high level models (for the log Repair, Loan and Hospital) output by our technique is shown in Table 5 and the basic information of the three high level logs created by our technique is shown in Table 6.

According to Table 6, the generated high level logs H-Repair and H-Hospital contains fewer activities than their related raw event logs Repair and Hospital. The main reason is that the activities in the original repair log and hospital log can form high quality activity clusters (more activity relations inside the cluster and fewer among the clusters). In the experiment about 1% events from log Hospital and 0.5% events from log Loan are removed together with the infrequent high level activities generated and for the log Repair no events are removed (very few events are removed because of the effects of the trace merging technique proposed in Section 3).

According to Table 5, all of the three high level models generated have high accuracy which benefits from the abstraction strategy put forward in Section 2. For the high level activities in the three built high level models, the average accuracy of their relevant sub-models is also generally good.

5 RELATED WORK

Trace clustering technique is one of the most effective approaches for dealing with the negative impacts from high variety of behaviors recorded in event logs. Several classical trace clustering approaches have been proposed in the literature. In (Song et al., 2009) the authors put forward an approach which is able to abstract the features of the traces from event logs into five profiles that includes *activity profile*, *transition profile*, *case attributes profile*, *event attributes profile* and *performance profile*. Afterwards, these profiles are converted into an aggregate vector so that the distance between any two traces can be measured. The main advantage of this technique is that it considers a complete range of metrics for clustering traces. In (Bose and van der Aalst, 2010) and (Bose and van der Aalst, 2009) the context-aware trace clustering techniques are proposed which try to improve the output results of trace clustering by employing the context knowledge that can be acquired from event logs. In (Bose and van der Aalst, 2010) the authors point out that the feature sets based on sub-sequences of traces are context-aware and can express some process functions. The traces that have many common conserved features should be put in the same cluster. The authors in (Bose and van der Aalst, 2009) develop an edit distance-based trace clustering algorithm. The context knowledge mined from event logs are integrated in the calculation procedure for the cost of edit operations. The Markov trace clustering method is put forward in (Ferreira et al., 2007). This method calculates a potential first-order Markov model for each cluster based on an expectation-maximization algorithm. A trace is sent to a cluster which has a Markov model that can generate this trace with a high probability. In (Weerdt et al., 2013) a novel technique named *active trace clustering* is presented. This technique tries to optimise the fitness of each cluster's underlying process model during the run time without employing the vector space model for the clustering process. It simply distributes the traces to the suitable clusters by considering the optimization of the combined accuracy of the potential models for these clusters. Most trace clustering techniques perform well for dealing with the event logs with a moderate amount of trace behaviors. However, such techniques can not assure a good result while being executed on the logs with massive behaviors (as is shown in the case study in Section 4).

Process model abstraction approach is also effective for dealing with the "spaghetti-like" business process models mined. In (Bose and van der Aalst, 2009) the authors develop a two-step approach for mining

Table 4: The weighted average quality of the sub-models generated by our method.

Log	$W_t - ICS$	$W_e - ICS$	$W_t - E{-}Cardoso$	$W_e - E{-}Cardoso$	$W_t - PT{-}CD$	$W_e - PT{-}CD$
Repair	0.9738	0.9687	11.57	12.46	2.0688	2.0929
Loan	0.9514	0.9297	21.934	26.4995	2.1729	2.2238
Hospital	0.8891	0.902	467.84	465.2	3.1257	3.0956

Table 5: The quality information of the high level models generated for each log by our technique.

Log	ICS	$E - Cardoso$	$PT{-}CD$
Repair	0.978	33	2.483
Loan	0.9671	137	3.378
Hospital	0.95	192	2.4328

Table 6: Basic information of the generated high level logs.

H-Log	**Traces**	**Events**	**Event types**
H-Repair	1000	2700	10
H-Loan	13087	40783	44
H-Hospital	1143	37740	65

hierarchical business process models. This approach searches for the sub-traces that repeatedly happen in event logs. Two kinds of such sub-traces are defined which are *tandem arrays* and *maximal repeats*. This approach firstly searches for all the *tandem arrays* and the *maximal repeats* in the event logs and then replace them in the original event logs by using high level activities (each high level activity is an abstraction of a *tandem array* or a *maximal repeat* found) so that the high level event logs can be generated. Finally, the high level models (more accurate and simpler) could be mined by using existing workflow discovery algorithms executed on the high level logs. The authors in (Baier and Mendling, 2013) indicate that the low level events recorded in the event logs may be too granular and should be mapped to the high level activities predefined in the enterprise process specifications. Hence, they put forward a mapping method that combines the domain knowledge captured from these specifications. With the high level activities generated the better models on the higher abstraction level can be built. The authors in (Conforti et al., 2014) present an automated technique for mining the BPMN models with subprocesses. This technique analyses the dependencies among the data attributes attached to events. The events that are judged to have high dependencies will be put in the same subprocesses. Most of the classical process model abstraction approaches presented focus mainly on searching for the subprocesses and can not assure the quality of the built high level models. It is possible that the high level activities in the underlying abstracted models may still have a large amount of relations among each other.

6 CONCLUSION

In this paper we proposed a new method which combines the characters of the classical model abstraction techniques and the trace clustering techniques for solving the problem of inaccurate and complex process models mined. This method is able to optimise the quality of the underlying high level models through an efficient abstraction strategy and also considers the quality of the sub-models generated through trace clustering techniques. Finally, the details of the business processes recorded in the event logs are revealed by the high level models built together with the generated sub-models where each sub-model shows the details of its relevant high level activity. Though the results of the case study we demonstrated the effectiveness of our technique.

Our future work will mainly be focused on developing new trace clustering techniques with higher performance to help deal with the complex and inaccurate sub-models generated for the high level activities. We will also validate our method on some other real-life cases.

REFERENCES

van der Aalst, W., van Dongen, B. F., Herbst, J., Maruster, L., Schimm, G. and Weijters, A. J. M. M. (2003). Workflow Mining: A Survey of Issues and Aproaches. In *Data and Knowledge Engineering*, 47(2): 237–267, 2003.

van der Aalst, W. (2011). *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer-Verlag Berlin Heidelberg, 1nd edition.

Weerdt, J. D., vanden Broucke, S., Vanthienen, J., and Baesens, B. (2013). Active Trace Clustering for Improved Process Discovery. *IEEE Transactions on Knowledge and Data Engineering*, 25(12):2708–2720.

Bose, R. and van der Aalst, W. (2009). Context Aware Trace Clustering: Towards Improving Process Mining Results. In *Proceedings of the SIAM International Conference on Data Mining*, pages 401–412.

Bose, R. and van der Aalst, W. (2010). Trace Clustering Based on Conserved Patterns: Towards Achieving Better Process Models. In *Business Process Management Workshops*, volume 43 of *Lecture Notes in Business Information Processing*, pages 170–181. Springer Berlin.

Song, M., Gnther, C. and van der Aalst, W. (2009). Trace Clustering in Process Mining. In *Business Process Management Workshops*, volume 17 of *Lecture Notes in Business Information Processing*, pages 109–120. Springer Berlin.

Ferreira, D. R., Zacarias, M., Malheiros, M.and Ferreira, P. (2007). Approaching Process Mining with Sequence Clustering: Experiments and Findings. In *Business Process Management (BPM 2007)*, volume 4714 of *Lecture Notes in Computer Science*, pages 360–374.

Bose, R. and van der Aalst, W. (2009). Abstractions in Process Mining: A Taxonomy of Patterns. In *Business Process Management (BPM 2009)*, volume 5701 of *Lecture Notes in Computer Science*, pages 159–175.

Baier, T. and Mendling, J. (2013). Bridging Abstraction Layers in Process Mining: Event to Activity Mapping. In *BPMDS 2013 and EMMSAD 2013*, volume 147 of *Lecture Notes in Business Information Processing*, pages 109–123.

Conforti, R., Dumas, M., Carcia-Banuelos, L. and Rosa, M. L. (2014). Beyond Tasks and Gateways: Discovering BPMN Models with Subprocesses, Boundary Events and Activity Markers. In *Business Process Management (BPM 2014)*, volume 8659 of *Lecture Notes in Computer Science*, pages 101–117.

Schaeffer, S. E. (2007). Graph Clustering. In *Computer Science Review*, 1(1):27–64, 2007.

Hompes, B. F. A., Verbeek, H. M. W. and van der Aalst, W. (2014). Finding Suitable Activity Clusters for Decomposed Process Discovery. In *Proc. of the 4th Int'l. Symp. on Data-driven Process Discovery and Analysis*, volume 1293 of *CEUR Workshop Proceeding*, pages 16–30.

Weijters, A., van der Aalst, W. and Alves de Medeiros, A. K. (2006). Process Mining with the Heuristics Algorithm. In *BETA Working Paper Series*, 166, 2006.

Noack, A. (2007). Energy Models for Graph Clustering. In *J. Graph Algorithms Appl*, 11(2):453–480, 2007.

de Medeiros, A. A. (2006). Genetic Process Mining. In *PhD. thesis, Eindhoven University of Technology*, 2006.

Lassen, K. B. and van der Aalst, W. (2009). Complexity Metrics for Workflow Nets. In *Inform. Software Technol.*, 51:610–626.