HAL
open science

Generalized Alignment-Based Trace Clustering of Process Behavior

Mathilde Boltenhagen, Thomas Chatain, Josep Carmona

To cite this version:

Mathilde Boltenhagen, Thomas Chatain, Josep Carmona. Generalized Alignment-Based Trace Clustering of Process Behavior. Petri Nets 2019 / ACSD 2019 - 40th International Conference on Application and Theory of Petri Nets and Concurrency, Jun 2019, Aachen, Germany. hal-02176771

HAL Id: hal-02176771

https://hal.science/hal-02176771v1

Submitted on 8 Jul 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Generalized Alignment-Based Trace Clustering of Process Behavior

Mathilde Boltenhagen[1], Thomas Chatain[1], and Josep Carmona[2]

[1] LSV, CNRS, ENS Paris-Saclay, Inria, Université Paris-Saclay, Cachan (France)
{boltenhagen,chatain}@lsv.fr
[2] Universitat Politècnica de Catalunya, Barcelona (Spain)
jcarmona@cs.upc.edu

Abstract. Process mining techniques use event logs containing real process executions in order to mine, align and extend process models. The partition of an event log into trace variants facilitates the understanding and analysis of traces, so it is a common pre-processing in process mining environments. Trace clustering automates this partition; traditionally it has been applied without taking into consideration the availability of a process model. In this paper we extend our previous work on process model based trace clustering, by allowing cluster centroids to have a complex structure, that can range from a partial order, down to a subnet of the initial process model. This way, the new clustering framework presented in this paper is able to cluster together traces that are distant only due to concurrency or loop constructs in process models. We show the complexity analysis of the different instantiations of the trace clustering framework, and have implemented it in a prototype tool that has been tested on different datasets.

1 Introduction

Process Mining is becoming an essential discipline to cope with the tons of process data arising in organizations [1]. Now an organization can use some of the available commercial tools to elicit and streamline its processes, so that its decisions are based on the evidences found in the data. In any of these existing software tools, the notion of *trace variant* is fundamental: it denotes a singular sequential execution of the process from start to end. All observed traces that correspond to the same permutation of activities (although the other data attributes, e.g., the customer name, are different), are included into the same trace variant. When trace variants are found, stakeholders then analyze them in order to find out possible incoherences between observed and modeled behavior [2].

In reality, however, the previous flow for analyzing process data is not as ideal as one may think. First, *event logs* that contain the process data stored by an organization, can contain noise, a phenomenon that affects the capability of identifying the right trace variant. Second, processes are not static entities in organizations, but instead evolve over time, which implies also a drift on the number and type of trace variants. Third, processes describing concurrent

behavior will tend to separate in different trace variants different interleavings of the same Mazurkiewicz trace, although perhaps the analysis for all these traces should be the same. The same applies in case of loops, where often it is not necessary to separate traces that only differ in the number of loop iterations.

In this paper we present a novel clustering technique that is able to tackle the aforementioned situations. Intuitively, the idea is to cluster the event log in a way that traces in the same cluster can be very distant when considered as words, but actually they correspond to the same trace variant when concurrency and loop behavior is disregarded. We build upon a technique presented in a recent paper [3], which assumes that a process model exists. This assumption is realistic in many contexts, e.g., in *Process-Aware Information Systems* (PAIS), process models are often available [4].

We extend the technique in [3] by allowing clustering centroids to now be partial-orders or even subnets of the process model. We present properties that relate them, and show how the subnet case can still be encoding in a SAT instance. Correspondingly, we adapt the notion of inter- and intra-cluster distance and spot quality criteria, so that a characterization of optimal clustering can be defined by users.

We see a great potential on the techniques presented in this paper: first, the techniques proposed can help into simplifying the analysis of event logs, by enabling a better (more abstract) characterization of trace variants. Second, the novel concurrency-and-loop-aware trace clustering proposed is significantly more robust than the ones found in the literature, and tends to avoid redundancy between different clusters.

Related work. Several techniques have been proposed in the last decade for trace clustering [5,6,7,8,9,10,11]. They can be partitioned into *vector space approaches* [5,7], *context aware approaches* [8,9] and *model-based approaches* [6,10,11]. All the aforementioned clustering algorithms consider only the event log as input, and use different internal representations for producing the clusters. In contrast, in a recent paper [3], we presented a different view on clustering event log traces, by assuming that a process model exists. All the aforementioned techniques do not allow concurrency or loop behavior.

The use of an explicit characterization of concurrency has been considered recently in process discovery: the works in [12,13] show how to improve the discovery of a process model by folding the initial unfolding that satisfies the independence relations given as inputs. In the area of conformance checking, the same phenomena has been observed: the work in [14] assumes traces are represented as partial order, thus allowing again an explicit characterization of concurrency in the problem formalization.

Perhaps the works more similar to the one of this paper are [15,16], where a transition system representing the event log is clustered, so that a set of simpler process models is generated. Tailored state-based properties that guarantee certain Petri net classes are used to guide the clustering, whereas in this work the computation of subnets is unrestricted.

Our work is also related to [17] which clusters events and detects deviation. However, our work focuses on an existing model and the results may consider different directions like repairs while [17] gives a pre-processing of data.

Complexity of our works is related to [18] which demonstrates several methods for distance between automata. Even for dynamic functions, the complexity have been proved PSPACE-complete and is even more complex for Petri nets which is formalism used in this paper.

Organization of the paper. In the next section we provide an example of the main contributions of the paper. Then, preliminaries are given in Section 3, and Section 4 defines the quality criteria for trace clustering. In Sections 5 and 6 we present the two main clustering perspectives proposed in this paper, and the complexity analysis of the problem of computing an optimal clustering is reported in Section 7. Finally, in Section 8 we provide an evaluation of the prototype implementation of the techniques of this paper over several event logs. Section 9 summarizes this paper and provides futures research lines.

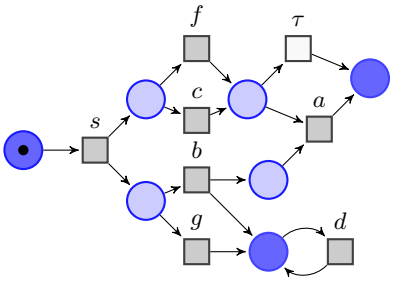
2 A Motivating Example

In [3], we introduced the idea of a trace clustering technique based on a known process model. Each group of traces is related to a trace variant, corresponding to a *full run* of the model (Def. 2), which serves as centroid. The traces in the cluster must all be sufficiently close to the centroid. This allows to identify executions of the model which reproduce typical observed traces, and also to isolate deviant log traces which are too far from what the model describes. In Fig. 1, we present an example of alignment-based clustering.

Our definitions deal with the distance between log traces and the centroid of their cluster. Since these are usually presented as words over an alphabet of actions, a notion of distance on words is used, typically Levenshtein's edit distance. Sometimes, concurrency and loop behavior is not important to differentiate two traces of a business process, as illustrated by the following example.

Example 1. Model of Fig. 1 describes the behaviors of users rating an app. First, users start the form (s). They give either a good (g) or a bad (b) mark attached to a comment (c) or a file (f). Bad ratings get apologies (a), a silent transition (τ) enables to avoid them. Finally, users can donate to the developers of the app (d). The company may be interested in grouping users by behavior, to visualize the differences; for instance which profiles provide bad marks. Trace clusterings of Fig. 1, 2 and 3 has been created, for a maximal distance of alignment to 1.

The order of concurrent actions, like writing a comment before or after giving the rating, does not need to matter to distinguish behaviors in this process. In the alignment-based trace clustering from [3], $\langle s, f, b, a\rangle$ and $\langle s, b, f, a\rangle$, of unhappy customers who uploaded a file, differ only on concurrent actions, and are separated in different clusters. In contrast, Fig. 2 shows a new trace clustering approach where concurrency is disregarded, with the consequence that the two



(a) Petri Net.

$\langle s, c, g \rangle$
$\langle s, c, g, d \rangle$
$\langle s, f, b, a \rangle$
$\langle s, f, f, a \rangle$
$\langle s, b, f, a \rangle$
$\langle s, g, f, d, d \rangle$
$\langle s, g, f, d, d, d, d \rangle$
$\langle g, c, f, s, d, d \rangle$
$\langle s, d, d, d \rangle$

(b) Log L_1

Centroids	Traces	Distance
$\langle s, c, \tau, g \rangle$	$\langle s, c, g \rangle$	0
	$\langle s, c, g, d \rangle$	1
$\langle s, b, f, a \rangle$	$\langle s, b, f, a \rangle$	0
	$\langle s, f, f, a \rangle$	1
$\langle s, f, b, a \rangle$	$\langle s, f, b, a \rangle$	0
$\langle s, g, f, \tau, d, d \rangle$	$\langle s, g, f, d, d \rangle$	0
$\langle s, g, f, \tau, d, d, d, d \rangle$	$\langle s, g, f, d, d, d, d \rangle$	0
non-clustered	$\langle g, c, f, s, d, d \rangle$	NA
	$\langle s, d, d, d \rangle$	NA

(c) Clusters

Fig. 1: Alignment-based Trace Clustering (ATC).

previous traces now belong to the same cluster. Underneath, the method uses partial-order runs (called processes) of the model instead of sequential runs, shown on the first column of Fig. 2.

Furthermore, donating twice or four times to the developers of the app represent very close behaviors and accordingly, similar profiles. Hence traces $\langle s, g, f, d, d \rangle$ and $\langle s, g, f, d, d, d, d \rangle$ should then be clustered in the same group. This is why we propose yet another trace clustering technique that allows for repetitive behavior in the same cluster, that uses subnets of the process model. Then the two traces below belong to a unique cluster, shown in Fig. 3.

Furthermore, aligning traces to the model is fundamental to avoid clustering (highly) deviant traces. For instance, the log trace $\langle g, c, f, s, d, d \rangle$ is left non-clustered in our work, but would be clustered with $\langle s, g, f, d, d \rangle$ for a trace clustering based only on the log. We compared the results of clusterings to [11] which grouped data by attributes frequency and occurrences, e.g. the activity names. Those traces are then groups with fitting traces.

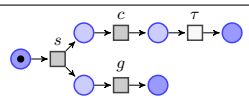
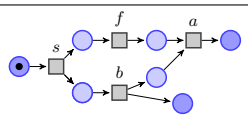
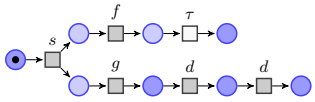
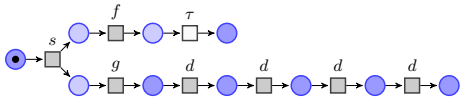
Centroids	Traces	Distance
	$\langle s, c, g \rangle$	0
	$\langle s, c, g, d \rangle$	1
	$\langle s, b, f, a \rangle$	0
	$\langle s, f, b, a \rangle$	0
	$\langle s, f, f, a \rangle$	1
	$\langle s, g, f, d, d \rangle$	0
	$\langle s, g, f, d, d, d, d \rangle$	0
non-clustered	$\langle g, c, f, s, d, d \rangle$	NA
	$\langle s, d, d, d \rangle$	NA

Fig. 2: Alignment and Partial Order based Trace Clustering (APOTC).

3 Preliminaries

3.1 Process Models and Trace Clustering

We assume process models are described as Petri nets [19]. Formally:

Definition 1 (Process Model (Labeled Petri Net)). *A Process Model defined by a* labeled Petri net system *(or simply* Petri net*) is a tuple* $N = \langle P, T, F, m_0, m_f, \Sigma, \lambda \rangle$*, where* P *is the set of places,* T *is the set of transitions (with* $P \cap T = \emptyset$*),* $F \subseteq (P \times T) \cup (T \times P)$ *is the flow relation,* m_0 *is the initial marking,* m_f *is the final marking,* Σ *is an alphabet of actions and* $\lambda : T \to \Sigma \cup \{\tau\}$ *labels every transition by an action or as silent.*

Semantics. The semantics of Petri nets is given in term of *firing sequences*. Given a node $x \in P \cup T$, we define its pre-set ${}^\bullet x \stackrel{\text{def}}{=} \{y \in P \cup T \mid (y, x) \in F\}$ and its post-set $x^\bullet \stackrel{\text{def}}{=} \{y \in P \cup T \mid (x, y) \in F\}$. A marking is an assignment of a non-negative integer to each place. A transition t is *enabled* in a marking m when all places in ${}^\bullet t$ are marked. When a transition t is enabled, it can *fire* by removing a token from each place in ${}^\bullet t$ and putting a token to each place in $t^\bullet$. A marking m' is *reachable* from m if there is a sequence of firings $\langle t_1 \dots t_n \rangle$ that transforms m into m', denoted by $m[t_1 \dots t_n \rangle m'$.

The set of reachable markings from m_0 is denoted by $[m_0\rangle$. A Petri net is *k-bounded* if no marking in $[m_0\rangle$ assigns more than k tokens to any place. A Petri net is *safe* if it is 1-bounded. In this paper we assume safe Petri nets.

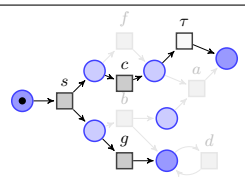
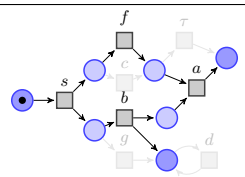
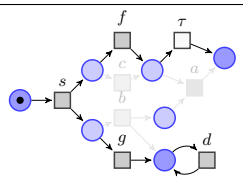
Centroids	Traces	Distance
	$\langle s, c, g \rangle$	0
	$\langle s, c, g, d \rangle$	1
	$\langle s, b, f, a \rangle$	0
	$\langle s, f, b, a \rangle$	0
	$\langle s, f, f, a \rangle$	1
	$\langle s, g, f, d, d \rangle$	0
	$\langle s, g, f, d, d, d, d \rangle$	0
non-clustered	$\langle g, c, f, s, d, d \rangle$	NA
	$\langle s, d, d, d \rangle$	NA

Fig. 3: Alignment and Model Subnet based Trace Clustering (AMSTC).

Definition 2 (Full Run). *A firing sequence* $u = \langle t_1 \dots t_n \rangle$ *such that* $m_0[u\rangle m_f$ *is called a* full run *of* N*. We denote by* $Runs(N)$ *the set of full runs of* N*.*

Given a full run $u = \langle t_1 \dots t_n \rangle \in Runs(N)$, the sequence of actions $\lambda(u) \stackrel{\text{def}}{=} \langle \lambda(t_1) \dots \lambda(t_n) \rangle$ is called a *(model) trace of* N. When the labeling function λ is injective, like in the model of Fig. 1, we sometimes identify the transition t with its label $\lambda(t)$. Then, full runs coincide with model traces. Examples for the model of Fig. 1 are $\langle s, c, g \rangle$, $\langle s, f, b, a \rangle$, $\langle s, f, g, d, d, d \rangle$.

Definition 3 (Log). *A* log *over an alphabet* Σ *is a finite set of words* $\sigma \in \Sigma^*$*, called* log traces*.*

Fig. 1c shows log traces of recorded behaviors.

Definition 4 (Trace Clustering). *Given a log* L*, a trace clustering over* L *is a partition over a (possibly proper) subset of the traces in* L*.*

Figures 1 and 2 show two different examples of trace clustering, with 5 and 4 clusters respectively.

Alignment-based trace clustering is a particular form of trace clustering: it relies on a model N of the observed system. The idea of alignment-based trace clustering is to explicit the relation between log traces and full runs of N. Concretely, each cluster of log traces will be assigned a full run u of N,

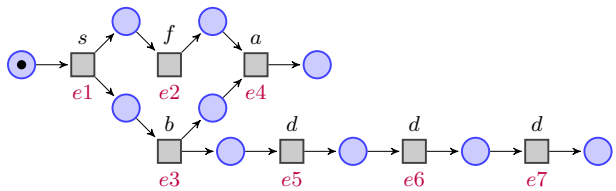


Fig. 4: Example of process of the Petri net in Fig. 1

presented as the centroid of the cluster. Hence, traces in the same cluster are not only similar among them, but they are related to a run of the model, which together validates a part of the model and explains the observed log traces.

Definition 5 (Alignment-based Trace Clustering (ATC) [3]). *For a log L and a Petri net $N = \langle P, T, F, m_0, m_f, \Sigma, \lambda \rangle$, an alignment-based trace clustering of L w.r.t. N is a tuple $\mathcal{C} = \langle \{u_1 \dots u_n\}, \chi \rangle$ where $u_1 \dots u_n$ ($n \in \mathbb{N}$) are full runs of N which serve as centroids for the clusters and $\chi : L \to \{\mathtt{nc}, u_1 \dots u_n\}$ maps log traces either to the centroid of its cluster $\chi(\sigma)$, or to none of the clusters, denoted by* $\mathtt{nc}$.

Each set $\chi^{-1}(u_i)$, for $i \in \{1 \dots n\}$, defines the cluster whose centroid is u_i. The set $\chi^{-1}(\mathtt{nc})$ contains the traces which are left non-clustered.

Fig. 1 shows a clustering of the traces of a log L_1 based on a model N. For this cluster, $\chi^{-1}(\langle s, g, c, \tau \rangle)$ is also the set of sequences: $\{\langle s, c, g \rangle, \langle s, g, c, d \rangle\}$. We remark that the traces $\langle g, c, f, s, d, d \rangle$ and $\langle s, d, d, d \rangle$ have not been classified for this clustering.

3.2 Partial-Order Semantics

In full runs of a process model, transition occurrences are totally ordered. However transitions can be handled in different orders for the same process in case of concurrency. In the model of Fig.1 traces $\langle s, b, f, a \rangle$ and $\langle s, f, b, a \rangle$ follow the same process but differ by the order of the transitions.

They can however be seen as two linearizations of a common representation based on partial-order runs which represents a *process*.

Definition 6 (Partial-Order Representation of Runs: Process). *A (non-branching) process $\mathcal{P}$ of a Petri Net $N = \langle P, T, F, m_0, m_f, \Sigma, \lambda \rangle$ is a tuple $\mathcal{P} = \langle B, E, G, B_0, B_f, h \rangle$ where:*

- *(B, E, G, B_0, B_f) is a non-branching, finite, acyclic Petri Net, i.e.*
  - *its causality relation G^+ is acyclic, and*
  - *it has no forward and no backward branchings:*

$$\forall b \in B \quad \exists! e \in E \cup \{\bot\} \quad b \in e^\bullet$$
$$\forall b \in B \quad \exists! e \in E \cup \{\top\} \quad b \in {}^\bullet e$$

where $\bot$ and $\top$ are virtual events satisfying $\bot^\bullet \stackrel{\text{def}}{=} B_0$ and ${}^\bullet\top \stackrel{\text{def}}{=} B_f$.

– $h : (B \cup E) \to (P \cup T)$ *is a function that maps the non-branching process* $\mathcal{P}$ *in the Petri Net* N *with the following relations:*
  - $h(B) \subseteq P$ *and* $h(E) \subseteq T$
  - $\forall e \in E, h_{|^\bullet e}$ *is a bijection between* $^\bullet e$ *and* $^\bullet h(e)$*, same reasoning for* $h_{|e^\bullet}$
  - $h_{|B_0}$ *is a bijection between* B_0 *and* m_0*, likewise for* $h_{|B_f}$

Fig. 4 shows a process of the Petri Net in Fig. 1. Each event e_i corresponds to a transition of N (for instance $h(e_2) = f$).

Definition 7 (Process representation of a full run). *Every full run* u *of a (safe) model* N *induces a process of* N*. This process is unique up to isomorphism [20] and is denoted by* $\Pi(u)$*.*

In general, a process represents several full runs, which differ only by the ordering of concurrent actions. For instance, both sequences $\langle s, f, b, a, d, d, d\rangle$ and $\langle s, b, d, d, d, f, a\rangle$ induce the process of the Fig. 4.

We write $Runs(\mathcal{P})$ for the set of full runs of the process $\mathcal{P}$. For every full run $\langle e_1 \dots e_n\rangle$ of a process $\mathcal{P}$ of a Petri net N, the sequence $u \stackrel{\text{def}}{=} \langle h(e_1) \dots h(e_n)\rangle \in T^*$ is called a *linearization* of $\mathcal{P}$. Every linearization of $\mathcal{P}$ is a full run of N.

3.3 Distances Between Log and Model Traces

A key element in this work, and in Process Mining in general, is to align log traces to full runs of the model. In this work, in particular, we target good alignment between every log trace $\sigma \in L$ and the centroid of its cluster $u = \chi(\sigma)$. By the labeling λ of transitions of the model, full runs are mapped to words over the alphabet Σ of actions, called model traces, and the quality of the alignment between σ and u can be quantified as the distance $dist(\sigma, \lambda(u))$, where $dist$ is a distance between finite words over Σ. In this paper, we use Levenshtein's edit distance, which is usually considered appropriate in Process Mining.

Definition 8 (Levenshtein's edit distance). *Levenshtein's edit distance* $dist(w1, w2)$ *between two words* w_1 *and* $w_2 \in \Sigma^*$ *is the minimal number of edits needed to transform* w_1 *to* w_2*. Editions can be substitutions to a letter by another one, deletions or additions of a letter in words.*

We will abuse notations, and write $dist(\sigma, u)$ for $dist(\sigma, \lambda(u))$, and $dist(u_1, u_2)$ for $dist(\lambda(u_1), \lambda(u_2))$. For example, the full run $\langle s, g, c\rangle$ and the log trace $\langle s, g, d, c\rangle$ have only one difference: the addition of d. They are at distance 1.

4 Quality Criteria for Trace Clustering

Fig. 5 shows two alignment-based trace clusterings of a new log L_2, based on the model N of Fig. 1. The two clusterings have been created for the same model and log, and contain different centroids.

In this section, we provide criteria which contribute in the qualification of a good clustering. We have identified the following criteria:

Clustering	Centroids u	Traces σ	$dist(\sigma, u)$	Quality criteria
$\mathcal{C}_1$	$\langle s, c, g, \tau \rangle$	$\langle s, f, g \rangle$	1	
		$\langle s, c, g \rangle$	0	$d(C_1) = 2$
		$\langle s, g, c \rangle$	2	
		$\langle s, c, g, d \rangle$	1	$\Delta(C_1) = 12$
	$\langle s, b, f, a \rangle$	$\langle s, f, b, a \rangle$	2	
		$\langle s, c, b, a \rangle$	2	$n(C_1) = 3$
		$\langle s, b, f, a \rangle$	0	
		$\langle s, b, c, a \rangle$	1	$\check{c}(C_1) = 0.83$
	$\langle s, f, \tau, g, d, d \rangle$	$\langle s, f, g, d \rangle$	1	
		$\langle s, f, g, d, d, d, d \rangle$	2	$\Phi(C_1) = 3$
	non-clustered	$\langle s, f, a, a, a \rangle$	NA	
		$\langle s, f, b, d, d \rangle$	NA	
$\mathcal{C}_2$	$\langle s, f, g, \tau \rangle$	$\langle s, f, g \rangle$	0	
		$\langle s, c, g, d \rangle$	2	$d(C_1) = 2$
		$\langle s, f, g, d \rangle$	1	
	$\langle s, f, b, a \rangle$	$\langle s, f, b, a \rangle$	0	$\Delta(C_2) = 14$
		$\langle s, f, a, a, a \rangle$	2	
		$\langle s, c, b, a \rangle$	1	$n(C_2) = 4$
		$\langle s, b, f, a \rangle$	2	
		$\langle s, b, c, a \rangle$	2	$\check{c}(C_2) = 1.0$
		$\langle s, f, b, d, d \rangle$	2	
	$\langle s, f, g, \tau, d, d, d, d \rangle$	$\langle s, f, g, d, d, d, d \rangle$	0	$\Phi(C_2) = 2$
	$\langle s, g, c, \tau \rangle$	$\langle s, c, g \rangle$	2	
		$\langle s, g, c \rangle$	0	

Fig. 5: Two possible clusterings for the same set of trace logs.

- $\boldsymbol{d(\mathcal{C})}$, *maximum distance between a trace and the centroid of its cluster*: this criterion, defined by $\max_{\sigma \in L \setminus \chi^{-1}(\mathtt{nc})} dist(\sigma, \chi(\sigma))$, will be minimized to increase the fit of the centroids to their traces. In case of a log containing noise, a small distance may induce many non-clustered traces.
- $\boldsymbol{\Delta(\mathcal{C})}$, *sum of distances* : the sum $\Delta(\mathcal{C}) \stackrel{\text{def}}{=} \sum_{\sigma \in L \setminus \chi^{-1}(\mathtt{nc})} dist(\sigma, \chi(\sigma))$ can be seen as a variant or a refinement of the previous criterion $d(\mathcal{C})$. It will also be minimized in order to get the most representative centroids.
- $\boldsymbol{n(\mathcal{C})}$, *number of clusters* : The number of clusters provides an interesting perspective, which is analogous to the number of trace variants of a process model, but in this case from the log perspective.
- $\boldsymbol{\check{c}(\mathcal{C})}$, *ratio of clustered traces:* this ratio, defined as $\check{c}(\mathcal{C}) \stackrel{\text{def}}{=} \frac{|L| - |\chi^{-1}(\mathtt{nc})|}{|L|}$, is close to 1 for a process model that covers most of the behavior of the log. $\check{c}(\mathcal{C})$ also highlights the ratio of distant traces, i.e. traces that deviate from the model, for a given maximum distance $d(\mathcal{C})$.
- $\boldsymbol{\Phi(\mathcal{C})}$, *inter-cluster distance* : the distance between the centroids is also an important parameter. For an ATC $\mathcal{C} = \langle \{u_1 \dots u_n\}, \chi \rangle$, the inter-cluster distance is defined as $\Phi(\mathcal{C}) \stackrel{\text{def}}{=} \min_{i \neq j} dist(u_i, u_j)$. A larger distance involves

distant clusters, this is why this parameter should be maximized in order to prevent overlay between the clusters.

Most of the detailed criteria come from the Data Mining domain [21,22]. Other measures like the Dunn [23], which compares distances between items that share or not a cluster, and the Silhouette [24], that computes if items is close enough to their clusters instead of the others, help the user to analyze its clustering. As usual when multiple parameters are taken into account, there will not exist in general a unique clustering optimizing all the criteria together. Instead, every clustering problem should consider a good balance between the parameters to optimize. Our tool, which is described in section 8, returns the optimal clustering for a given pre-defined setting.

Example 2. Fig 5 shows two ATCs of the model N of Fig. 1 and a new set of log traces L_2. The results differ on the parameters to optimize. The first one minimize the inter-cluster distance $\Phi(\mathcal{C}_1)$ for a given distance between the trace and the centroid $d(\mathcal{C}_1)$ to 2. However, some traces are left non-clustered which do not appear in the second clustering. In contrast, the centroids are closer ($\Phi(\mathcal{C}_2) < \Phi(\mathcal{C}_1)$) and the number of clusters is larger ($n(\mathcal{C}_1) < n(\mathcal{C}_2)$).

5 Fitting Centroids to Concurrency

The aim of ATC is to group traces which are similar to a full run of the model. In this section, we want to go further and allow one to cluster together traces which differ only by the order of execution of transitions which are presented as concurrent in the model. In the ATC of Fig. 1, traces $\langle s, b, f, a\rangle$ and $\langle s, f, b, a\rangle$ are clustered separately, and since no model trace is at distance ≤ 1 to both of them, every ATC $\mathcal{C}$ which would cluster them together would have an inter-cluster $d(\mathcal{C}) > 1$. Yet, $\langle s, b, f, a\rangle$ and $\langle s, f, b, a\rangle$ are perfectly aligned with two different interleavings of *the same* execution of the model, if one understands "execution" as process like in Def. 6. The following definition of trace clustering, precisely uses processes as cluster centroids.

Definition 9 (Alignment and Partial Order based Trace Clustering (APOTC)). *As full run clustering, an alignment and partial order based trace clustering, of a log L and a Petri net $N = \langle P, T, F, m_0, m_f, \Sigma, \lambda\rangle$, is a tuple $\mathcal{C} = \langle\{\mathcal{P}_1 \dots \mathcal{P}_n\}, \chi\rangle$ where $\mathcal{P}_1 \dots \mathcal{P}_n$ ($n \in \mathbb{N}$) are processes of N which serve as centroids for the clusters and $\chi : L \to \{\mathtt{nc}, \mathcal{P}_1 \dots \mathcal{P}_n\}$ maps log traces either to the centroid of its cluster $\chi(\sigma)$, or to none of the clusters, denoted by* $\mathtt{nc}$.

5.1 Quality Criteria for APOTC

All the quality criteria of ATC are considered in APOTC, but they need to be redefined now that centroids are processes. Indeed we need to compare log traces to processes, which represent (finite) sets of full runs. Naturally, the distance between a model trace σ and a process $\mathcal{P}$ will be defined as the distance to its closest linearization of $\mathcal{P}$.

Definition 10. *We define the distance $dist(\sigma, \mathcal{P})$ (abusing notation dist again) between a trace σ and a process $\mathcal{P}$ as $dist(\sigma, \mathcal{P}) \stackrel{\text{def}}{=} \min_{u \in Runs(\mathcal{P})} dist(\sigma, u)$.*

This allows us to define $d(\mathcal{C})$ and $\Delta(\mathcal{C})$ for APOTC like for ATC, respectively as $\max_{\sigma \in L \setminus \chi^{-1}(\texttt{nc})} dist(\sigma, \chi(\sigma))$ and $\sum_{\sigma \in L \setminus \chi^{-1}(\texttt{nc})} dist(\sigma, \chi(\sigma))$ with $\chi(\sigma)$ are processes.

The inter-cluster distance of an APOTC $\mathcal{C} = \langle \{\mathcal{P}_1 \dots \mathcal{P}_n\}, \chi \rangle$ is also defined as for ATC, as the minimum distance between two centroids: $\Phi(\mathcal{C}) = \min_{i \neq j} dist(\mathcal{P}_i, \mathcal{P}_j)$, using the appropriate notion of distance between processes:

Definition 11. *The distance between two processes is the minimal distance between their linearizations: $dist(\mathcal{P}, \mathcal{P}') \stackrel{\text{def}}{=} \min_{\substack{u \in Runs(\mathcal{P}) \\ u' \in Runs(\mathcal{P}')}} dist(u, u')$.*

Example 3. Fig. 2 shows an APOTC of the model and log of Fig. 1. Traces $\langle s, b, f, a \rangle$ and $\langle s, f, b, a \rangle$ can now be clustered together, yielding a smaller $n(\mathcal{C})$ for equivalent $\Delta(\mathcal{C})$ and $d(\mathcal{C})$.

5.2 Relating APOTC to ATC

Any ATC can be casted as an APOTC. All the full runs centroids of an ATC, which are sequential executions, can be represented as processes using Def. 7. The following theorem explains how this transformation affects the quality criteria of the clusterings.

Theorem 1. *For any ATC $\mathcal{C}_u = \langle \{u_1 \dots u_n\}, \chi_u \rangle$, we define $\forall i \in \{1 \dots n\}$ $\mathcal{P}_i \stackrel{\text{def}}{=} \Pi(u_i)$ and $\chi_{\mathcal{P}} \stackrel{\text{def}}{=} \Pi \circ \chi_u$ (by convention $\Pi(\texttt{nc}) = \texttt{nc}$) inducing $\mathcal{C}_{\mathcal{P}} = \langle \{\mathcal{P}_1 \dots \mathcal{P}_n\}, \chi_{\mathcal{P}} \rangle$ its corresponding APOTC of the same process model N and the same log L. The distances below follow the properties:*

1. *$d(\mathcal{C}_u) \geq d(\mathcal{C}_{\mathcal{P}})$ and $\Delta(\mathcal{C}_u) \geq \Delta(\mathcal{C}_{\mathcal{P}})$ with equality if the model is sequential*
2. *$\Phi(\mathcal{C}_u) \geq \Phi(\mathcal{C}_{\mathcal{P}})$ with equality if the model is sequential*
3. *$n(\mathcal{C}_u) = n(\mathcal{C}_{\mathcal{P}})$ and $\check{c}(\mathcal{C}_u) = \check{c}(\mathcal{C}_{\mathcal{P}})$*

Proof. We first observe that the obtained set $\{\mathcal{P}_1 \dots \mathcal{P}_n\}$ is by Def. 7 a set of subnets of N and $\chi_{\mathcal{P}}$ maps every clustered log traces to a subnet and non-clustered log traces to $\texttt{nc}$. Then $\mathcal{C}_{\mathcal{P}} = \langle \{\mathcal{P}_1 \dots \mathcal{P}_n\}, \chi_{\mathcal{P}} \rangle$ is indeed an APOTC.

1. Every trace σ of L is either clustered ($\chi_u(\sigma) = u_i,\ i \in \{1 \dots n\}$) or non-clustered ($\chi_u(\sigma) = \texttt{nc}$). The maximum distance between traces and centroids $d(\mathcal{C}_u)$ depends only on clustered traces: $\forall \sigma \in L_{\setminus \chi_u{}^{-1}(nc)} \quad dist(\sigma, \chi_u(\sigma)) \leq d(\mathcal{C}_u)$. By Def. 7 $\chi_u(\sigma) \in Runs(\chi_{\mathcal{P}}(\sigma))$. Then for any clustered trace σ, we have $d(\mathcal{C}_{\mathcal{P}}) \leq dist(\sigma, \chi_{\mathcal{P}}(\sigma)) \leq dist(\sigma, \chi_u(\sigma)) \leq d(\mathcal{C}_u)$ with equality if the model is sequential (no other run in $Runs(\chi_{\mathcal{P}}(\sigma))$). Furthermore, $\Delta(\mathcal{C})$ is the sum of the distances: $\Delta(\mathcal{C}_{\mathcal{P}}) \leq \Delta(\mathcal{C}_u)$.
2. Let u_i and u_j, $i, j \in \{1 \dots n\}$, be two centroids of the ATC. The corresponding processes of those centroids are defined by $\mathcal{P}_i = \Pi(u_i)$ and

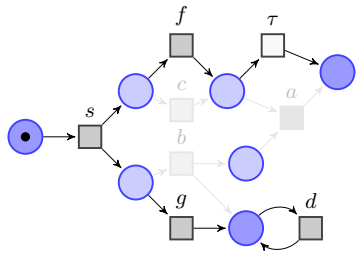


Fig. 6: A subnet of the Petri Net in Fig. 1. Only transitions s, g, f, τ, d are kept. Transitions in light gray do not belong to the subnet.

$\mathcal{P}_j = \Pi(u_j)$ and $u_i \in Runs(\mathcal{P}_i)$ and $u_j \in Runs(\mathcal{P}_j)$. This implies $dist(\mathcal{P}_i, \mathcal{P}_j) \leq dist(u_i, u_j)$ with equality if the model is sequential (no other run in the processes). Consequently $\Phi(\mathcal{C}_\mathcal{P}) \leq \min_{i \neq j} dist(u_i, u_j) = \Phi(\mathcal{C}_u)$ with equality if the model is sequential.

3. This is immediate by definition of $\chi_\mathcal{P}$. □

As a summary, casting an ATC to an APOTC improves the distances between traces and centroids; in contrast, the resulting APOTC may get a lower (i.e. poorer) inter-cluster distance than the ATC. The number of clusters and ratio of clustered traces are preserved.

This means that clusters that were distant in the ATC may become closer in the APOTC, which appears negative when seen from the perspective of good clusterings presenting distant clusters. But, in the other hand, clusters that become closer will typically be those that one precisely wanted to merge because they represent different interleavings of processes. This is exactly what happens in Example 3. Merging clusters then results in a lower number of clusters $n(\mathcal{C})$, which also helps to get a human understandable clustering and facilitates the analysis of the results by decision makers.

Example 4. When casting the ATC of Fig. 1 to an APOTC, the clusters with centroid $\langle s, b, f, a \rangle$ and $\langle s, f, d, a \rangle$ become two clusters with the same process as centroid. This leads to an inter-cluster distance $\Phi(\mathcal{C}_\mathcal{P}) = 0$ for the APOTC. But, after merging these two clusters, one gets the better APOTC presented in Fig. 2.

6 Fitting Centroids to Concurrency and Repetitive Behavior

In Fig. 2, we show that APOTC separates process arising from traces corresponding to different number of loop iterations, e.g., the traces $\langle s, g, f, d, d \rangle$ and $\langle s, g, f, d, d, d, d \rangle$. The issue is due of the finite size of runs of processes. Indeed process centroids are partial order runs which do not allow loops and infinite sequences of events. To overcome this limitation, we introduce subnets of models.

Definition 12 (Subnet of Petri net). *A* subnet *of a Petri net* $N = \langle P, T, F, m_0, m_f, \Sigma, \lambda \rangle$ *is a Petri net* $\langle P, T', F_{|T'}, m_0, m_f, \Sigma_{|T'}, \lambda \rangle$ *with* $T' \subseteq T$, *and* $F_{T'} \stackrel{\text{def}}{=} F \cap (P \times T' \cup T' \times P)$.

Fig. 6 presents a subnet of the model of Fig. 1. Observe that our definition of subnets, based on selecting transitions, restricts the semantics of the net and cannot produce new behaviors. Formally:

Lemma 1. *Every full run (resp. process) of a subnet of a Petri net* N*, is a full run (resp. process) of* N.

We now formalize AMSTC, which consider subnets as centroids:

Definition 13 (Alignment and Model Subnet-based Trace Clustering (AMSTC)). *For a log* L *and a Petri net* $N = \langle P, T, F, m_0, m_f, \Sigma, \lambda \rangle$*, an alignment and model subnet trace clustering, of* L *w.r.t.* N *is a tuple* $\mathcal{C} = \langle \{\mathcal{N}_1 \ldots \mathcal{N}_n\}, \chi \rangle$ *where* $\mathcal{N}_1 \ldots \mathcal{N}_n$ *are subnets of* N *which serve as centroids for the clusters and* $\chi : L \to \{\texttt{nc}, \mathcal{N}_1 \ldots \mathcal{N}_n\}$ *maps log traces either to the centroid of its cluster* $\chi(\sigma)$*, or to none of the clusters, denoted by* $\texttt{nc}$.

Fig. 3 shows an AMSTC of log traces based on the model of Fig. 1.

6.1 Quality Criteria for AMSTC

All the previous criteria for ATC and APOTC also apply to AMSTC, but the notion of distance needs to be adapted again. The quality criteria $d(\mathcal{C})$ and $\Delta(\mathcal{C})$ rely on the distance between log traces and the centroids of their clusters. Here, centroids are subnets, and the distance is defined as $dist(\sigma, \mathcal{N}) \stackrel{\text{def}}{=} \min_{u \in Runs(\mathcal{N})} dist(\sigma, u)$. Computing this distance corresponds to aligning the trace to the model.

The inter-cluster distance, $\Phi(\mathcal{C})$ is the minimal distance between two subnet centroids, now defined as: $dist(\mathcal{N}, \mathcal{N}') \stackrel{\text{def}}{=} \min_{\substack{u \in Runs(\mathcal{N}) \\ u' \in Runs(\mathcal{N}')}} dist(u, u')$.

Example 5. Fig. 3 shows an AMSTC of the Petri Net and the log traces of Fig. 1 for $d(\mathcal{C}) = 1$. Then the traces $\langle s, g, f, d, d \rangle$ and $\langle s, g, f, d, d, d, d \rangle$ are grouped in the same cluster.

Intra-cluster distance. If applied unrestricted, AMSTC can use as centroids, subnets with branchings and loops, and then cluster together very different log traces. The intra-cluster distance aims at controlling this aspect. For instance, taking as centroid the complete net of Fig. 1 would not yield a satisfactory AMSTC. Instead, traces in the same cluster should be similar and represent a generalized notion of trace variant. This criterion is quantified by the *intra-cluster distance* $\Theta(\mathcal{C})$. Clusterings with low $\Theta(\mathcal{C})$ will be preferred.

- $\boldsymbol{\Theta(\mathcal{C})}$, *the intra-cluster distance*: Before defining the intra-cluster distance of a clustering $\mathcal{C}$, we focus on each of its centroids separately: for every centroid $\mathcal{N}_k$, define

$$\Theta'(\mathcal{N}_k) \stackrel{\text{def}}{=} \sup_{\mathcal{P},\mathcal{P}' \in Proc(\mathcal{N}_k)} \frac{dist(\mathcal{P},\mathcal{P}')}{(1+\epsilon)^{max(|\mathcal{P}|,|\mathcal{P}'|)}}$$

where $|\mathcal{P}|$ denotes the number of events in $\mathcal{P}$, and $\epsilon > 0$ is a parameter set by the user in order to limit (more or less) the influence of long processes. Indeed, when the subnet $\mathcal{N}_k$ has loops, it has infinitely many processes, arbitrary large, which yields arbitrary large distance to the smaller processes. Yet, such subnets may be relevant, as illustrated by Example 6. This is why our definition penalizes more for distances between small processes.

Finally, the intra-cluster distance of a clustering $\mathcal{C} = \langle\{\mathcal{N}_1 \dots \mathcal{N}_n\}, \chi\rangle$ is:

$$\Theta(\mathcal{C}) \stackrel{\text{def}}{=} \max_k \Theta'(\mathcal{N}_k)$$

Example 6. The AMSTC of Fig. 3 has only one centroid with a loop. Because of the loop d, this centroid has infinitely many processes. With $\epsilon = 0.1$, the intra-cluster distance of this AMSTC is bounded by 2.39, increasing ϵ to 0.5 returns $\Theta(\mathcal{C}) = 0.12$ which penalizes significantly less the loop in the subnet centroid.

6.2 Relating AMSTC to APOTC

Every APOTC $\mathcal{C}_\mathcal{P}$ induces an AMSTC whose subnet centroids are subnets are defined according to the process centroids of $\mathcal{C}_\mathcal{P}$.

Definition 14 (Subnet induced by a process). *Every process $\mathcal{P} = (B, E, G, B_0, B_f, h)$ of a model $N = \langle P, T, F, m_0, m_f, \Sigma, \lambda\rangle$, induces a subnet of N defined by $\Psi(\mathcal{P}) \stackrel{\text{def}}{=} (P, h(E), G_{|h(E)}, h(B_0), h(B_f))$.*

Fig. 6 shows the subnet corresponding to the last process of Fig. 2. All the transitions in the process belong to the subnet, and the loop fits traces with arbitrary repetition of activity d (for instance $\langle s, f, g, d, d\rangle$ and $\langle s, f, g, d, d, d, d\rangle$).

The following theorem relates APOTC and the induced AMSTC, analogously to Theorem 1 for ATC and APOTC.

Theorem 2. *For any APOTC $\mathcal{C}_\mathcal{P} = \langle\{\mathcal{P}_1 \dots \mathcal{P}_n\}, \chi_{\mathcal{P}_i}\rangle$, we define $\forall i \in \{1 \dots n\}$ $\mathcal{N}_i \stackrel{\text{def}}{=} \Psi(\mathcal{P}_i)$ and $\chi_{\mathcal{N}_\mathcal{P}} \stackrel{\text{def}}{=} \Psi \circ \chi_\mathcal{P}$ (by convention $\Psi(\texttt{nc}) = \texttt{nc}$) inducing $\mathcal{C}_{\mathcal{N}_\mathcal{P}} = \langle\{\mathcal{N}_1 \dots \mathcal{N}_n\}, \chi_{\mathcal{N}_\mathcal{P}}\rangle$ its corresponding AMSTC of the same process model N and the same log L. The distances below follow the properties:*

1. *$d(\mathcal{C}_\mathcal{P}) \geq d(\mathcal{C}_{\mathcal{N}_\mathcal{P}})$ and $\Delta(\mathcal{C}_\mathcal{P}) \geq \Delta(\mathcal{C}_{\mathcal{N}_\mathcal{P}})$ with equality if the model is acyclic*
2. *$\Phi(\mathcal{C}_\mathcal{P}) \geq \Phi(\mathcal{C}_{\mathcal{N}_\mathcal{P}})$ with equality if the model is acyclic.*
3. *$n(\mathcal{C}_\mathcal{P}) = n(\mathcal{C}_{\mathcal{N}_\mathcal{P}})$ and $\check{c}(\mathcal{C}_\mathcal{P}) = \check{c}(\mathcal{C}_{\mathcal{N}_\mathcal{P}})$*

Proof. The correspondance APOTC-AMSTC is similar to the correspondance ATC-APOTC demonstrated in Theorem 1. When properties exactly coincide between ATC and APOTC for sequential models, same results are found between APOTC and AMSTC for acyclic models: without loops, every subnet has a single process and the distances are preserved. □

As subnets may allow infinite runs, the sum of differences $\Delta(\mathcal{C})$ between log traces and centroids may decrease in an expansion from APOTC to AMSTC. Unfortunately, the inter-cluster distance $\Phi(\mathcal{C})$ can also be lower.

When AMSTCs meet APOTCs. Observe that, in our definition of AMSTC, only the behavior of the subnets is considered. Hence, nothing penalizes a clustering for having dead transitions in a cluster, i.e., transitions which do not participate in any full run of the subnet. Intuitively, this situation is not satisfactory since we expect the subnets to give information about the part of the net which really participates in the observed traces. By the way, notice that the subnets induced by processes following Def. 14 never have any dead transition. These subnets also have another property: they all have at least one full run. Let us call *fair* an AMSTC in which every centroid has these two properties. The following theorem establishes a relation between APOTCs and fair AMSTCs.

Theorem 3. *For a log L and an acyclic and trace-deterministic*[3] *model N, the transformation defined in Theorem 2 establishes a bijection from the set of APOTC to the set of fair AMSTCs $\mathcal{C}$ with intra-cluster distance $\Theta(\mathcal{C}) = 0$.*

Proof. Since N is acyclic, for every process $\mathcal{P}$ of N, the subnet induced by $\mathcal{P}$ has no other process than $\mathcal{P}$ itself. This proves that any AMSTC $\mathcal{C}$ obtained from an APOTC has intra-cluster distance $\Theta(\mathcal{C}) = 0$. It is also fair as we noticed earlier.

Now, every centroid $\mathcal{N}_i$ of a fair AMSTC $\mathcal{C}$ with $\Theta(\mathcal{C}) = 0$ has a single process (call it $\mathcal{P}_i$): indeed, since the model is trace-deterministic, every subnet centroid in $\mathcal{C}$ having two different processes would lead to $\Theta(\mathcal{C}) > 0$. This establishes a bijection between the centroids of fair AMSCs with intra-cluster distance 0, and the processes of N, which serve as centroids in APOTCs. This bijection between centroids induces naturally our bijection between APOTCs and AMSTCs. □

In summary, AMSTC handles both concurrency and repetitive behavior, and under some situations behaves similarly to APOTC.

7 Complexity of Alignment-based Trace Clusterings

For a log L and a model N, one is typically interested in computing a trace clustering (ATC, APOTC or AMSTC) $\mathcal{C}$ of L w.r.t. N of sufficient quality, i.e. satisfying some constraints on the quality criteria $d(\mathcal{C})$, $\Delta(\mathcal{C})$, $n(\mathcal{C})$, $\check{c}(\mathcal{C})\ldots$ We will see that, at least from a theoretical point of view, the complexity lies already in the existence of a clustering, and the specification of many quality constraints does not change the complexity.

For a non-empty log L and a model N, there exists an ATC $\mathcal{C}$ of L w.r.t. N having $\check{c}(\mathcal{C}) > 0$ (i.e. such that at least one trace is clustered), iff N has a full run. Indeed, when no constraint is given about the quality criteria $d(\mathcal{C})$, $\Delta(\mathcal{C})$, $n(\mathcal{C})$, $\Phi(\mathcal{C})\ldots$, any full run of N can serve as centroid, and any log trace can be assigned to any cluster. The same holds for APOTC, where centroids are processes of N, since N has a process iff N has a full run; it holds again for AMSTC, taking into account the constraint that any subnet used as centroid should have a full run, or the stronger constraint that the subnet should not have any dead transition, as discussed in Section 6.2.

[3] N is *trace-deterministic* if the mapping $u \in Runs(N) \mapsto \lambda(u) \in \Sigma^*$ is injective.

Now, deciding if a model has a full run u, corresponds to checking reachability of the final marking. The problem of reachability in Petri nets is known to be decidable, but non-elementary [25], and still PSPACE-complete for safe Petri nets. But the complexity trivially drops to NP-complete[4] if a bound l is given (with l an integer coded in unary) on the length of u.

In practice, relevant clusterings will not use very long full runs (or processes for APOTC) as centroids. Also for AMSTC, no very long full run will be considered in the computation of $d(\mathcal{C})$, $\Delta(\mathcal{C})$ or $\Phi(\mathcal{C})$. Typically, a bound l on the length of the full runs can be assumed, for instance 2 times the length of the longer log trace. Let us call l-bounded a trace clustering satisfying this constraint.

Theorem 4. *The problem of deciding, for a log L, a model N, an integer bound l, integers $d_{\max}$, $\Delta_{\max}$, $n_{\max}$ and a rational number $\check{c}_{\min}$, the existence of a l-bounded ATC (respectively APOTC, AMSTC) $\mathcal{C}$ of L w.r.t. N, having $d(\mathcal{C}) \leq d_{\max}$, $\Delta(\mathcal{C}) \leq \Delta_{\max}$, $n(\mathcal{C}) \leq n_{\max}$ and $\check{c}(\mathcal{C}) \geq \check{c}_{\min}$, is NP-complete.*

Proof. As observed earlier, the problem is NP-hard even with the only constraint that at least one trace is clustered (i.e. $\check{c}(\mathcal{C}) > 0$, or equivalently $\check{c}(\mathcal{C}) \geq \frac{1}{|L|}$). It remains to show that it is in NP: indeed, if there exists a (l-bounded) clustering, there exists one with no more that $|L|$ clusters (forgetting empty clusters cannot weaken the quality criteria); and, by assumption, the size of centroids (defined as $|\sigma|$ for ATC, $|\mathcal{P}|$ for APOTC, number of transitions in the subnet for AMSTC) is bounded by l. So, it is possible to guess a clustering $\mathcal{C}$ in polynomial time. For APOTC and AMSTC, one can also guess in P time the full run $u \in \mathit{Runs}(\chi(\sigma))$, for every clustered trace σ, which will achieve the $\mathit{dist}(\sigma, \chi(\sigma))$. Now, checking that $\mathcal{C}$ satisfies the constraints, only requires to compute Levenshtein's edit distances and minima over sets of polynomial size. This can be done in P time. □

For ATC, the problem remains in NP with an additional constraint on the inter-cluster distance ($\Phi(\mathcal{C}) \geq \Phi_{\min}$) because the inter-cluster-distance can be computed in P time.

On the other hand, incorporating new constraints like bounds on $\Phi(\mathcal{C})$ for APOTC or AMSTC, or on the intra-cluster distance $\Theta(\mathcal{C})$, may increase the complexity. The principle of the algorithm remains: guess non-deterministically a clustering, then check if it satisfies the constraints. Hence, the complexity depends on the complexity of the algorithm used as an oracle to check, given a log, a model and a clustering $\mathcal{C}$, if $\mathcal{C}$ satisfies the constraints. Precisely, if there exists such an oracle algorithm in some complexity class A, then the l-bounded trace clustering problem is in NP^{A}. For instance, for APOTC, checking if $\Phi(\mathcal{C}) \geq \Phi_{\min}$ is in NP; in consequence, the trace clustering problem with such constraint is in $\mathrm{NP}^{\mathrm{NP}}$. We get the same result for constraints on the intra-cluster distance ($\Theta(\mathcal{C}) \geq \Theta_{\min}$) for AMSTC.

[4] NP-hardness can be obtained by reduction from the problem of reachability in a safe acyclic Petri net, known to be NP-complete [26,27].

8 SAT Encoding and Experimentation

The NP-completeness established in Theorem 4 for our trace clustering problems suggests to encode them as SAT problems. For each clustering problem, our tool DarkSider constructs a pseudo-Boolean[5] formula and calls a solver (currently minisat+ [28]). Every solution to the formula is interpreted as a trace clustering. This is already what we did for a version of ATC presented in our previous paper [3], to which we refer the reader for a more detailed description of the SAT encoding. Here, we present the main ideas for the encoding of AMSTC, which is done in the same spirit[6].

The assignment of log traces to clusters in an AMSTC $\mathcal{C} = \langle \{\mathcal{N}_1 \dots \mathcal{N}_n\}, \chi \rangle$ is encoded using variables $(\chi_{\sigma k})_{\sigma \in L,\, k=1\dots n}$ meaning that $\chi(\sigma) = \mathcal{N}_k$. Variables $(c_{kt})_{k=1\dots n,\, t \in T}$ code the fact that transition t appears in subnet $\mathcal{N}_k$.

In order to encode that a sequence $u = \langle t_1 \dots t_n \rangle$ is a full run of N (or of a subnet $\mathcal{N}_k$), we use a set of Boolean variables:

- $\tau_{i,t}$ for $i = 1 \dots n$, $t \in T$: means that transition $t_i = t$; and
- $m_{i,p}$ for $i = 0 \dots n$, $p \in P$: means that place p is marked in marking m_i reached after firing $\langle t_1 \dots t_i \rangle$ (remind that we consider only safe nets, therefore the $m_{i,p}$ are Boolean variables).

They are involved in constraints like, for instance:

- Initial marking: $\big(\bigwedge_{p \in m_0} m_{0,p}\big) \wedge \big(\bigwedge_{p \in P \setminus m_0} \neg m_{0,p}\big)$
- Transitions are enabled when they fire: $\bigwedge_{i=1}^{n} \bigwedge_{t \in T} (\tau_{i,t} \implies \bigwedge_{p \in {}^\bullet t} m_{i-1,p})$.

Finally, variables $\delta_{\sigma i}$ are used to detect and count the mismatches between a (clustered) log trace $\sigma \in L$ and the (closest) execution of the subnet $\chi(\sigma)$ which serves as centroid for its cluster. These variables are needed to encode the constraints $d(\mathcal{C}) \leq d_{\max}$ and $\Delta(\mathcal{C}) \leq \Delta_{\max}$, or to construct a minimization objective for the solver when one wants to find AMSTC which minimize these quantities.

As explained in Section 7, dealing with constraints about the inter-cluster and intra-cluster distance pushes our AMSTC problem out of the NP complexity class. Concretely, this means that such constraints are not adapted for a SAT encoding. Instead, we use an approximation of the inter-cluster distance, by bounding the number of common transitions between centroids, and the intra-cluster distance, by bounding the number of transition per centroids.

8.1 Experimental results

Our tool DarkSider[7] implements the computation of ATCs and AMSTCs and optimizes the inter-cluster distance and the proportion of clustered traces.

[5] Pseudo-Boolean constraints are generalizations of Boolean constraints. They allow one to specify constant bounds on the number of variables which can/must be assigned to true among a set V of variables.

[6] Due to AMSTC being the most general trace clustering, our experiments focus on this method.

[7] https://github.com/BoltMaud/darksider

Model			$\|L\|$	Clustering	Formulas size		Execution	$d(\mathcal{C})$	$n(\mathcal{C})$	$\Phi(\mathcal{C})$	$\check{c}(\mathcal{C})$
Reference	$\|T\|$	$\|P\|$			Variables	Constraints	Time (sec)				
Fig. 1	8	7	12	ATC	13854	26457	0.66	2	3	3	1.0
Fig. 1	8	7	12	ATC	13854	26457	0.50	0	3	2	0.25
Fig. 1	8	7	12	AMSTC	27306	51897	1.52	2	2	2	1.0
Fig. 1	8	7	12	AMSTC	27306	51897	1.14	0	3	1	0.83
Fig. 1	8	7	300	ATC	348800	641530	1252.53	2	3	2	0.91
Fig. 1	8	7	300	AMSTC	868592	1641844	1449.14	2	2	4	0.99
subnet of M1 [29]	17	14	12	ATC	98924	218568	9.18	2	3	1	0.38
subnet of M1 [29]	17	14	12	AMSTC	191876	418462	15.76	2	2	5	0.54
subnet of M1 [29]	17	14	100	ATC	433491	925574	124.57	2	3	3	0.80
subnet of M1 [29]	17	14	100	AMSTC	1185839	2573419	5659.75	2	2	5	0.95
subnet of M1 [29]	17	14	100	AMSTC	1185839	2573419	AO:100.16	2	2	5	0.95
M1 [29]	40	40	12	ATC	297176	678664	108.30	3	2	5	0.41
M1 [29]	40	40	12	AMSTC	713769	1656046	88.27	3	2	5	0.5
M1 [29]	40	40	100	ATC	692247	1385046	AO: 309.50	3	3	3	0.80
M1 [29]	40	40	100	AMSTC	4978835	11559283	AO: 230.12	3	3	3	0.83

Table 1: Experimental results for the computation of ATC and AMSTC with our tool DarkSider, obtained on a virtual machine with CPU Intel®Core i5-530U-1.8GHz*2 and 5.2GB RAM.
AO (almost optimal) indicates experiments where we stopped the SAT solver before it finds an *optimal* solution; this way, we got much better execution times and still very satisfactory solutions.

We firstly experimented the clusterings of the model in Fig. 1 and the logs of Fig. 5. We increased the log size and the model size to observe the limits of finding optimal solutions of our alignment-based trace clustering problems. Log traces are slightly noisy, to show different kinds of solutions. As the computation of $\Delta(\mathcal{C})$ takes time due to the numerous combinations for the pseudo-SAT encoding, this criterion has been removed for testing various sizes entries. Likewise, for the complexity reasons explained in Section 7, our tool does not implement the optimization of the inter-cluster distance $\Phi(\mathcal{C})$; it simply computes it a posteriori.

Table 1 shows experimental results. Our encoding automatically gets the minimal number of required clusters which is usually a parameter to set [22]. Notice that our tool computes *optimal* clusterings for a given setting w.r.t. the inter-cluster distance. Of course, this quest of optimality is very expensive in computation time. This is why, for the larger problems, we stopped the solver after it found solutions that we considered close to the optimal. These experiments are labeled AO in Table 1. For the model with 17 transitions and 14 places and 100 traces, this dramatically decreased the execution time from 5659.75 to 100.16 seconds.Furthermore, for larger logs and larger models, like 100 traces and the model M1 of [29], we stopped the computation of the optimum, however we got almost optimal results with a ratio of clustered traces to 0.83, where the optimum would be 0.95.

9 Conclusion and Future Work

In this work, we have investigated novel alignment-based trace clustering techniques, that generalize the notion of centroid in different directions: concurrency and repetitive behavior. The paper proposes quality criteria for characterizing trace clustering, and adapts them for each one of the new instantiations proposed. Also, the situations where the two different instantiations collapse are described formally. Furthermore, a complexity analysis on the different instantiations of the methods is reported. The approach has been implemented and tested over some datasets, showing that it can be applied in practice.

As future work we have many avenues to follow: first, we plan to investigate the SAT encoding and interaction with the SAT solver, so that a better performance can be attained. Second, we plan to explore applications of the theory of this paper; we see several possibilities in different process mining sub-domains, ranging from concept drift, down to predictive monitoring.

Acknowledgments. This work has been supported by Farman institute at ENS Paris-Saclay and by MINECO and FEDER funds under grant TIN2017-86727-C2-1-R.

References

1. van der Aalst, W.M.P.: Process Mining - Data Science in Action, Second Edition. Springer (2016)
2. Carmona, J., van Dongen, B.F., Solti, A., Weidlich, M.: Conformance Checking - Relating Processes and Models. Springer (2018)
3. Chatain, T., Carmona, J., van Dongen, B.F.: Alignment-based trace clustering. In: Conceptual Modeling - 36th International Conference, ER 2017, Proceedings. (2017) 295–308
4. Dumas, M., van der Aalst, W.M.P., ter Hofstede, A.H.M.: Process-Aware Information Systems: Bridging People and Software Through Process Technology. Wiley (2005)
5. Greco, G., Guzzo, A., Pontieri, L., Saccà, D.: Discovering expressive process models by clustering log traces. IEEE Trans. Knowl. Data Eng. **18**(8) (2006) 1010–1027
6. Ferreira, D.R., Zacarias, M., Malheiros, M., Ferreira, P.: Approaching process mining with sequence clustering: Experiments and findings. In: Business Process Management, 5th International Conference, BPM 2007, Proceedings. 360–374
7. Song, M., Günther, C.W., van der Aalst, W.M.P.: Trace clustering in process mining. In: Business Process Management Workshops, BPM 2008 International Workshops, Milano, Italy, September 1-4, 2008. Revised Papers. (2008) 109–120
8. Bose, R.P.J.C., van der Aalst, W.M.P.: Context aware trace clustering: Towards improving process mining results. In: Proceedings of the SIAM International Conference on Data Mining, SDM 2009. (2009) 401–412
9. Bose, R.P.J.C., van der Aalst, W.M.P.: Trace clustering based on conserved patterns: Towards achieving better process models. In: Business Process Management Workshops, BPM 2009 International Workshops, Revised Papers. (2009) 170–181
10. Weerdt, J.D., vanden Broucke, S.K.L.M., Vanthienen, J., Baesens, B.: Active trace clustering for improved process discovery. IEEE Trans. Knowl. Data Eng. **25**(12) (2013) 2708–2720

11. Hompes, B., Buijs, J., van der Aalst, W., Dixit, P., Buurman, H.: Discovering deviating cases and process variants using trace clustering. In: Proceedings of the 27th Benelux Conference on Artificial Intelligence (BNAIC 2015). (2015)
12. Ponce de León, H., Rodríguez, C., Carmona, J., Heljanko, K., Haar, S.: Unfolding-based process discovery. In: Automated Technology for Verification and Analysis - 13th International Symposium, ATVA 2015, Proceedings, Springer (2015) 31–47
13. Ponce de León, H., Rodríguez, C., Carmona, J.: POD - A tool for process discovery using partial orders and independence information. In: Proceedings of the BPM Demo Session 2015 Co-located with the 13th International Conference on Business Process Management (BPM 2015). (2015) 100–104
14. Lu, X., Fahland, D., van der Aalst, W.M.P.: Conformance checking based on partially ordered event data. In: Business Process Management Workshops - BPM 2014 International Workshops, Eindhoven, Revised Papers. (2014) 75–88
15. de San Pedro, J., Cortadella, J.: Mining structured Petri nets for the visualization of process behavior. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing. (2016) 839–846
16. Mokhov, A., Cortadella, J., de Gennaro, A.: Process windows. In: 17th International Conference on Application of Concurrency to System Design, ACSD 2017. (2017) 86–95
17. Lu, X., Fahland, D., van den Biggelaar, F.J., van der Aalst, W.M.: Detecting deviating behaviors without models. In: International Conference on Business Process Management, Springer (2016) 126–139
18. Benedikt, M., Puppis, G., Riveros, C.: Regular repair of specifications. In: Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science, LICS 2011. (2011) 335–344
19. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE **77**(4) (April 1989) 541–574
20. Engelfriet, J.: Branching processes of Petri nets. Acta Informatica **28**(6) (1991) 575–591
21. Gonzalez, T.F.: Clustering to minimize the maximum intercluster distance. Theoretical Computer Science **38** (1985) 293–306
22. Berkhin, P.: A survey of clustering data mining techniques. In: Grouping multidimensional data. Springer (2006) 25–71
23. Dunn, J.C.: A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. (1973)
24. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics **20** (1987) 53–65
25. Czerwinski, W., Lasota, S., Lazic, R., Leroux, J., Mazowiecki, F.: The reachability problem for Petri nets is not elementary (extended abstract). CoRR **abs/1809.07115** (2018)
26. Stewart, I.A.: Reachability in some classes of acyclic Petri nets. Fundam. Inform. **23**(1) (1995) 91–100
27. Cheng, A., Esparza, J., Palsberg, J.: Complexity results for 1-safe nets. Theor. Comput. Sci. **147**(1&2) (1995) 117–136
28. Eén, N., Sörensson, N.: Translating pseudo-boolean constraints into SAT. JSAT **2**(1-4) (2006) 1–26
29. Taymouri, F., Carmona, J.: Model and event log reductions to boost the computation of alignments. In: Proceedings of the 6th International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2016). (2016) 50–62