Alignment-Based Trace Clustering

Thomas Chatain, Josep Carmona, Boudewijn Van Dongen

Alignment-Based Trace Clustering

Thomas Chatain, Josep Carmona, and Boudewijn van Dongen

LSV, ENS Paris-Saclay, CNRS, INRIA, Cachan (France)
chatain@lsv.ens-cachan.fr
Universitat Politècnica de Catalunya, Barcelona (Spain)
jcarmona@cs.upc.edu
Eindhoven University of Technology, The Netherlands
b.f.v.dongen@tue.nl

Abstract. A novel method to cluster event log traces is presented in this paper. In contrast to the approaches in the literature, the clustering approach of this paper assumes an additional input: a process model that describes the current process. The core idea of the algorithm is to use model traces as centroids of the clusters detected, computed from a generalization of the notion of *alignment*. This way, model explanations of observed behavior are the driving force to compute the clusters, instead of current model agnostic approaches, e.g., which group log traces merely on their vector-space similarity. We believe alignment-based trace clustering provides results more useful for stakeholders. Moreover, in case of *log incompleteness*, *noisy logs* or *concept drift*, they can be more robust for dealing with highly deviating traces. The technique of this paper can be combined with any clustering technique to provide model explanations to the clusters computed. The proposed technique relies on encoding the individual alignment problems into the (pseudo-)Boolean domain, and has been implemented in our tool DarkSider that uses an open-source solver.

1 Introduction

The ubiquity of digital data has made organizations to become more than ever data-oriented. This has a clear implication on the way decisions are taken in an organization, where nowadays an unprecedent focus is put to the evidences hidden in the data. *Process mining* is an emerging field which focuses on analyzing *event logs* which contain the data corresponding to process executions. Process mining techniques focus on discovering, analyzing and enhancing evidence-based process models [1].

Trace clustering has been used as a method to partition event logs in a way that more homogeneous sublogs are obtained, with the hope that process discovery techniques will perform better on the sublogs than if applied to the original log [1]. Several techniques have been proposed in the last decade for trace clustering [2,3,4,5,6,7,8]. They can be partitioned into *vector space approaches* [2,4], *context aware approaches* [5,6] and *model-based approaches* [3,7,8]. All the aforementioned clustering algorithms consider only the event log as input, and use different internal representations for producing the clusters.

We present a different view on clustering event log traces, by assuming 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 [9]. Notice that due to the evolving nature of processes, this assumption by no means invalidates the motivation of this work: processes in a organization evolve and/or change frequently, and therefore process mining (and consequently, clustering) techniques may be very useful to be aligned with the reality, even if a process model exists.

Most of the aforementioned algorithms for trace clustering are centroid-based, i.e., for each cluster a representative (often a vector of features) is the reference of the cluster when computing distances. Furthermore, in some algorithms this representative may not be one of the log traces (e.g., if applying *k-means*). By using these model agnostic approaches, the grouping of event log traces may have no relation at all with the executions of the underlying process.

The approach we propose in this paper puts the available process model as a first citizen in trace clustering: clusters computed have as centroid a process model execution. This way, even in case of deviations, incomplete or noisy traces, or even drifts in the process model (e.g., dealing with the process of winter sales, while the traces correspond to summer sales), a process explanation of the traces in each cluster is available, so that stakeholders can relate them more reliably to the underlying process.

The clustering approach of this paper has as core operation the novel concept of *multi-alignments*, which is also a contribution of the paper. Multi-alignments are a generalization of the notion of *alignments* [10]. Intuitively, given a trace representing a real process execution, an optimal alignment provides the best trace the process model can provide to reproduce the observed behavior. Then observed and model traces are rendered in a two-row matrix denoting the synchronous/asynchronous moves between individual activities of model and log, respectively. Multi-alignments generalize alignments in that not one but a collection of observed traces is considered, while still the model produces a single trace that globally aligns well (i.e., its at minimal global distance) with the observed traces. Multi-alignments are shown graphically as an $(n + 1)$-row matrix, where the first n rows correspond to the observed traces, and the $n + 1$ row denotes the model trace. An example of multi-alignment for four observed traces and a model with behavior according to the regular expression[1] $A; ((B1; C1)||(B2; C2)||(B3; C3)); D$ is shown in Figure 1.

The clustering approach of this paper is guided by the computation of multi-alignments. More specifically, clusters in our algorithm are multi-alignments. Informally, given a threshold distance, the approach produces a set of multi-alignments that covers (if possible) the set of traces in the event log. Several variations of the initial algorithm proposed in this paper can be envisioned as future work, e. g., hierarchical or density based, or which allow rising the given distance to guarantee a full covering of the traces in the event log.

[1] Operators ; and || denote sequential and parallel composition, respectively.

A	$B1$				$C1$	$B2$	$C2$	$B3$	$C3$					D	trace 1
A		$B2$	$C2$	$B1$	$C1$			$B3$	$C3$					D	trace 2
A								$B3$	$C3$	$B1$	$C1$	$B2$	$C2$	D	trace 3
A	$B1$	$B2$	$C2$		$C1$			$B3$	$C3$					D	trace 4
A	$B1$	$B2$	$C2$		$C1$			$B3$	$C3$					D	model trace

Fig. 1. A model trace which is an optimal multi-alignment for four log traces.

The paper is organized as follows: Section 2 provides the necessary background for the understanding of the paper. Then in Section 3 the formal description and algorithmic computation of multi-alignments is provided. The overall method for alignment-based clustering is presented in Section 4. The evaluation of the techniques is reported in Section 5. Finally, Section 6 concludes the paper and provides lines for future research.

2 Preliminaries

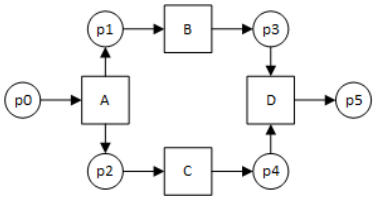
2.1 Petri Nets

Definition 1 ((Labeled) Petri Net System). *A* labeled Petri net system *(or simply* Petri net*) [11] is a tuple* $N = \langle P, T, \mathcal{F}, m_\perp, m_\top, \Sigma, \lambda \rangle$*, where* P *is the set of places,* T *is the set of transitions (with* $P \cap T = \emptyset$*),* $\mathcal{F} \subseteq (P \times T) \cup (T \times P)$ *is the flow relation,* $m_\perp$ *is the initial marking,* $m_\top$ *is the final marking,* Σ *is an alphabet of actions and* $\lambda : T \to \Sigma$ *labels every transition by an action.*

A marking is an assignment of a non-negative integer to each place. If k is assigned to place p by marking m (denoted $m(p) = k$), we say that p is marked with k tokens. Given a node $x \in P \cup T$, we define its pre-set ${}^\bullet x \stackrel{\text{def}}{=} \{y \in P \cup T \mid (x, y) \in F\}$ and its post-set $x^\bullet \stackrel{\text{def}}{=} \{y \in P \cup T \mid (y, x) \in F\}$.

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 $m[t_1 \dots t_n \rangle m'$. A firing sequence $u = \langle t_1 \dots t_n \rangle$ is called a *run* if it can fire from the initial marking: $m_\perp[u\rangle$; it is called a *full run* if it additionally reaches the final marking: $m_\perp[u\rangle m_\top$. We write $Runs(N)$ for the set of full runs of Petri net N. Given $u = \langle t_1 \dots t_n \rangle \in Runs(N)$, the sequence of actions $\lambda(u) \stackrel{\text{def}}{=} \langle \lambda(t_1) \dots \lambda(t_n) \rangle$. is called a *trace* of N.

The set of reachable markings from $m_\perp$ is denoted by $[m_\perp\rangle$, and form a graph called *reachability graph*. A Petri net is *k-bounded* if no marking in $[m_\perp\rangle$ 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.



(a) A model.

A	C	B	D
>>	C	>>	D

(b) An optimal alignment.

Fig. 2. Example of alignment between observed and modeled behavior.

2.2 Foundations of Alignments

We survey definitions for alignments and some variations. The interested reader can refer to [10] for the seminal work on alignments where a complete formalization can be found.

An event log is a collection of traces, where a trace may appear more than once. Formally:

Definition 2 (Event Log). *An* event log *L (over an alphabet of actions Σ) is a multiset of traces $\sigma \in \Sigma^*$.*

Given a Petri net N (typically obtained using process mining techniques, and supposed to model the behavior of an observed system), and an observed trace σ in a log, the aim of alignments is to find the full run u of the model N that mostly resembles σ, i.e. such that $\lambda(u)$ is close to σ, for some notion of distance $dist(\sigma, \lambda(u))$.

Example 1. An example of alignment is shown in Fig. 2: given the model in Fig. 2(a) and the trace $\langle C, D \rangle$, the model produces the trace $\langle A, C, B, D \rangle$, as shown in the upper row of Fig. 2(b).

A traditional choice for the distance $dist(\sigma, \gamma)$ is Levenshtein's edit distance (which counts how many deletions and insertions are needed to transform σ to γ). Another possible choice is Hamming distance, which simply counts the number of positions in which σ and γ differ: for two traces $\sigma = \langle \sigma_1 \dots \sigma_n \rangle$ and $\gamma = \langle \gamma_1 \dots \gamma_n \rangle$ of equal length n, Hamming distance is defined as $\left|\left\{i \in \{1 \dots n\} \mid \gamma_i \neq \sigma_i\right\}\right|$; when one trace is shorter than the other, we pad it and count every occurrence of the padding symbol as a mismatch with the longer trace.

Most of the definitions in this article are valid for any choice of distance (in particular Levenshtein and Hamming). For readability, some of the most technical developments are done only for Hamming distance, and we give insights on how to adapt them to Levenshtein's distance. In the example of Fig. 2, the trace produced by the model is at distance 2, independently of the distance considered. Our tool DarkSider mostly uses Levenshtein's distance.

Definition 3 (δ-Alignment, (Optimal) Alignment). *Given a (log) trace σ, and a Petri net model N, a* δ-alignment *of σ to N is a full run u of N such that $dist(\sigma, \lambda(u)) \leq \delta$.*

Clearly there exist δ-alignments for all values of δ larger or equal to a minimal value which we denote $\delta_{\min}(\sigma, N)$ (or simply $\delta_{\min}$ when σ and N are clear from the context). An optimal alignment *(or simply* alignment*) is a $\delta_{\min}(\sigma, N)$-alignment of σ to N.*

In Example 1, the 2-alignment represented by the run $\langle A, C, B, D\rangle$ is provided for the observed trace $\langle A, C\rangle$, which is an optimal alignment.

3 Multi-Alignments

We now present multi-alignment as a generalization of the notion of alignments. This new notion will be used in the rest of the paper as basis for the clustering approach proposed. We refer the reader to the example in the introduction (Figure 1) for an example of multi-alignment.

3.1 Formalization of Multi-alignments

The following definition relies on a notion of distance *dist*, which can be chosen depending on the context. For instance, Hamming distance and Levenshtein's edit distance are valid choices (see discussion in Section 2.2).

Definition 4 (Multi-alignments, Optimal Multi-alignments). *Given a finite collection $\mathcal{C}$ of (log) traces, a model N and some $\delta \in \mathbb{N}$, a* δ-multi-alignment *of $\mathcal{C}$ to N is a full run $u \in Runs(N)$ such that*[2] $\sum_{\sigma \in \mathcal{C}} dist(\sigma, \lambda(u)) \leq \delta$. *Clearly, there is a minimal δ for which a δ-multi-alignment exists. We denote it by $\delta_{\min}(\mathcal{C}, N)$ (or simply by $\delta_{\min}$ when $\mathcal{C}$ and N are clear from the context). A $\delta_{\min}$-multi-alignment is simply called an optimal multi-alignment of $\mathcal{C}$ to N.*

Given a log L, interesting instantiations deserve a comment. First, it is clear that if $|\mathcal{C}| = 1$, then the notion of multi-alignment collapses into the traditional notion of alignment from [10]. If $\mathcal{C} = L$, then the corresponding optimal multi-alignment represents the model trace that aligns optimally with all the traces in the log. However, for models containning alternative executions, to consider $\mathcal{C} = L$ may incur into high multi-alignment distances.

3.2 Encoding Multi-alignments Using Pseudo-Boolean Constraints

Computing multi-alignments is an NP-complete problem.[3]

[2] We understand the $\sum$ as a sum over a multiset, taking multiplicities into account. For instance, with the multiset $A = \{1, 1\}$, we get $\sum_{i \in A} i = 2$.

[3] More precisely, the problem of existence of a δ-multi-alignment for given $\mathcal{C}$, N and δ (represented in unary), is NP-complete. For NP-hardness, we use a reduction from the problem of reachability of a marking m in a 1-safe acyclic Petri net N, known to be NP-complete [12,13], to the existence of a 0-multi-alignement with the empty collection $\mathcal{C} = \emptyset$.

In order to compute a multi-alignment of $\mathcal{C}$ to N, our tool DarkSider constructs a pseudo-Boolean[4] formula $\Phi(N, \mathcal{C})$ and calls a solver (currently minisat+ [14]) to find an optimal solution. Every optimal solution to the formula is interpreted as a multi-alignment.

The formula $\Phi(N, \mathcal{C})$ characterizes a δ-multi-alignment $u = \langle t_1 \dots t_n \rangle \in \mathit{Runs}(N)$, with $\delta = \sum_{\sigma \in \mathcal{C}} \mathit{dist}(\sigma, \lambda(u))$. For simplicity, we present the encoding for Hamming distance as *dist*. Later we discuss how to adapt the encoding to Levenshtein's edit distance.

For the encoding, we need to fix a bound on the length $n = |u|$ of the δ-multi-alignment. In principle n could be exponential in $|T|$, simply because multi-alignments are full runs and there are models for which the final marking is reachable only after firing sequences of exponential length in $|T|$. Nevertheless, the distance $\mathit{dist}(\sigma, \lambda(u))$ between a full run u of length n is at least $n - |\sigma|$.[5] Hence u cannot be a δ-multi-alignment for δ smaller than $n - \min_{\sigma \in \mathcal{C}} |\sigma|$. Since in practice we are interested with δ of the order of the length of the log traces, we can bound n to, for instance, twice the length of the longest trace in $\mathcal{C}$: $n = 2 \times \max_{\sigma \in \mathcal{C}} |\sigma|$.

The formula $\Phi(N, \mathcal{C})$ is coded using the following Boolean variables:

- $\tau_{i,t}$ for $i = 1 \dots n$, $t \in T$ means that transition $t_i = t$.
- $m_{i,p}$ 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).
- $\delta_{i,\sigma}$ for $i = 1 \dots n$, $\sigma \in \mathcal{C}$, means that σ and $\lambda(u)$ mismatch at position i, i.e. $\lambda(t_i) \neq \sigma_i$.

The total number of variables is smaller than $n \times (|T| + |P| + |\mathcal{C}|)$.

Let us decompose the formula $\Phi(N, \mathcal{C})$.

- The fact that $u = \langle t_1 \dots t_n \rangle \in \mathit{Runs}(N)$ is coded by the conjunction of the following formulas:
  - Initial marking:

$$\left(\bigwedge_{p \in m_\perp} m_{0,p} \right) \wedge \left(\bigwedge_{p \in P \setminus m_\perp} \neg m_{0,p} \right)$$

  - Final marking:

$$\left(\bigwedge_{p \in m_\top} m_{n,p} \right) \wedge \left(\bigwedge_{p \in P \setminus m_\top} \neg m_{n,p} \right)$$

[4] 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. We write them as $a \leq \sum_{v \in V} v \leq b$. Pseudo-Boolean constraints are not more expressive but can be up to exponentially more concise than Boolean constraints. Some pseudo-Boolean solvers also offer to search for a solution minimizing a pseudo-Boolean objective of the same form $\sum_{v \in V} v$: number of variables assigned to true among V.

[5] This holds as well for Hamming or edit distance.

- The transitions are enabled when they fire:

$$\bigwedge_{i=1}^{n} \bigwedge_{t \in T} (\tau_{i,t} \implies \bigwedge_{p \in {}^\bullet t} m_{i-1,p})$$

- Token game (for safe Petri nets):

$$\bigwedge_{i=1}^{n} \bigwedge_{t \in T} \bigwedge_{p \in t^\bullet} (\tau_{i,t} \implies m_{i,p})$$

$$\bigwedge_{i=1}^{n} \bigwedge_{t \in T} \bigwedge_{p \in {}^\bullet t \setminus t^\bullet} (\tau_{i,t} \implies \neg m_{i,p})$$

$$\bigwedge_{i=1}^{n} \bigwedge_{t \in T} \bigwedge_{p \in P, p \notin {}^\bullet t, p \notin t^\bullet} (\tau_{i,t} \implies (m_{i,p} \iff m_{i-1,p}))$$

– One and only one t_i for each i, can be expressed concisely as a pseudo-Boolean constraint:

$$\sum_{i=1}^{n} \sum_{t \in T} \tau_{i,t} = 1$$

– Now, we look for a solution minimizing the quantity $\delta = \sum_{\sigma \in \mathcal{C}} dist(\sigma, \lambda(u))$ (total number of mismatches), which is coded as:

$$\sum_{\sigma \in \mathcal{C}} \sum_{i=1}^{n} \delta_{i,\sigma}$$

with the $\delta_{i,\sigma}$ correctly affected w.r.t. $\lambda(t_i)$ and σ_i:

$$\bigwedge_{\sigma \in \mathcal{C}} \bigwedge_{i=1}^{n} \left(\delta_{i,\sigma} \iff \bigvee_{t \in T,\ \lambda(t) = \sigma_i} \tau_{i,t} \right)$$

Notice that, as such, the formula $\Phi(N, \mathcal{C})$ characterizes multi-alignments u of a fixed length n. But in fact, what we need is to accept any u of length less or equal to n. There is a simple trick for this: we simply add to the Petri net N a new transition with the final marking $m_\top$ as pre- and post-set, and labeled with a special padding symbol. This transition allows N to 'wait' once in the final marking.

Size of the Formula. In the end, the first part of the formula ($u = \langle t_1 \dots t_n \rangle \in \mathit{Runs}(N)$) is coded by a pseudo-Boolean formula of size $O(n \times |T| \times |P|)$.

The second part of the formula (minimization of the mismatches) is coded by a pseudo-Boolean minimization objective of size $O(n \times |\mathcal{C}| \times |T|)$.

The total size for the coding of the formula $\Phi(N, \mathcal{C})$ is

$$O\left(n \times |T| \times \left(|P| + |\mathcal{C}|\right)\right) .$$

Encoding of Levenshtein's Edit Distance. The encoding that we have presented above is for multi-alignments w.r.t. Hamming distance (see discussion in Section 2.2). For Levenshtein's edit distance, the basic idea is to let the multi-alignment model trace "wait" for the log traces. This corresponds to the blanks in the matrix shown in Figure 1. For this it suffices to add to the Petri net N a new transition with empty pre- and post-set, and labeled with a special padding symbol. When counting the mismatches between the multi-alignment trace $\lambda(u)$ and a model trace σ, a blank compared to another action costs 1 (it corresponds to a deletion if one transforms σ to $\lambda(u)$). Symmetrically, the log trace is also allowed to "wait" for the multi-alignment.

There is now a subtlety: since the multi-alignment is compared with several log traces together, at some point it may have to wait for only some of the traces. This should count for the computation of the distance between $\lambda(u)$ and these traces, but not between $\lambda(u)$ and the other traces which do not need to wait. The solution is to let the latter wait without counting any mismatch at this point between them and $\lambda(u)$. This situation happens at the end of the multi-alignment shown in Figure 1 when the multi-alignment has to wait for trace 3, while trace 1, 2 and 4, which "are ready", wait like the multi-alignment.

In the end, this may lengthen the representation of the multi-alignment: in the worst case, for log traces of length l, one may have to insert $l - 1$ blanks before each symbol of the multi-alignment.

3.3 Partial Covering of the Log Traces

We have defined a multi-alignment as a full run u of the model N which minimizes its distance to a collection $\mathcal{C}$ of (log) traces. Now, if the collection $\mathcal{C}$ contains very different traces, it makes sense to focus on a subset of $\mathcal{C}$ containing sufficiently similar traces. For this we adapt a little bit the notion of multi-alignment in order to leave the choice of a subset of $\mathcal{C}$ to be considered: instead of minimizing the sum of distances to the log traces $\sigma \in \mathcal{C}$, we fix a distance threshold d and look for a $u \in \mathit{Runs}(N)$ which maximizes the number of log traces which are at distance $\leq d$ to u.

Definition 5 (Optimal Partial Covering). *Given a collection $\mathcal{C}$ of (log) traces, a model N and a distance threshold $d \in \mathbb{N}$, we say that a full run $u \in \mathit{Runs}(N)$ of N* covers *a log trace $\sigma \in \mathcal{C}$ if $\mathit{dist}(\sigma, \lambda(u)) < d$. We say that u is an* optimal partial covering *of $\mathcal{C}$ for the distance threshold d if no full run of N covers strictly more log traces of $\mathcal{C}$ than u.*

As an example, consider the following collection of log traces, the model shown in Figure 3, and set the distance threshold to $d = 4$.

```
t1: A C E B D G H F
t2: A C E G H F D B
t3: A C AC AA AF AI AJ AD AH X2 AG AB D B
t4: A E C G H F D B
t5: A C AA AC AF AJ AI AD AH X2 AG AB D B
```

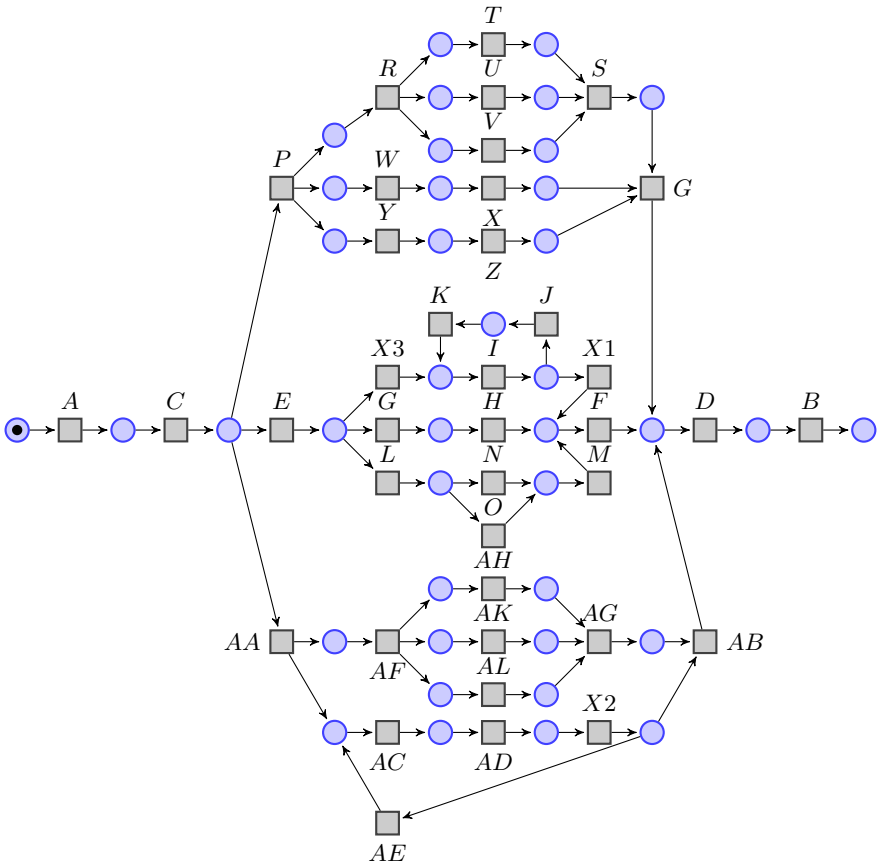


Fig. 3. M1 example (taken from [15]).

The full run $u_1 = \langle A, C, E, G, H, F, D, B\rangle$ covers traces `t1`, `t2` and `t4`. The full run $u_2 = \langle A, C, AA, AC, AF, AI, AJ, AD, AH, X2, AG, AB, D, B\rangle$ covers the others. No other full run covers more traces, so u_1 is an optimal partial covering.

The problem of finding an optimal partial covering u of the log traces can be encoded as a pseudo-Boolean optimization problem following the same lines as the encoding presented for multi-alignments in Section 3.2. We use additional variables b_σ for $\sigma \in \mathcal{C}$ with the constraint that b_σ can be assigned to true only if σ is covered by the full run u:

$$b_\sigma \implies \sum_{i=1}^{n} \delta_{i,\sigma} \leq d\,.$$

Now we express our objective of covering as many log traces as possible, as the pseudo-Boolean maximization objective $\sum_{\sigma\in\mathcal{C}} b_\sigma$.

4 Alignment-Based Clustering of Log Traces

Based on partial multi-alignments, we propose a novel algorithm for trace clustering. The idea of our algorithm is to group log traces according to their closeness to representative full runs of a given model. Those representative full runs act as centroids for the clusters.

The following algorithm partitions a collection $\mathcal{C}$ of log traces into a set P of clusters relying on a model N. Each cluster contains traces sufficiently close (i.e. at distance $\leq d$) to a full run $u \in \mathit{Runs}(N)$ which is the centroid of the cluster.

Clustering algorithm *Clustering*($\mathcal{C}$, N, d)

- $P := \emptyset$
- **repeat**
  - find a full run u of N which is an optimal partial covering of $\mathcal{C}$ for the distance threshold d;
  - let $C = \{\sigma \in \mathcal{C} \mid \mathit{dist}(\sigma, u) \leq d\}$ be a new cluster with u as centroid.
  - $P := P \cup \{C\}$
  - $\mathcal{C} := \mathcal{C} \setminus C$

  until all the traces are clustered or no remaining log trace $\sigma \in \mathcal{C}$ is at distance smaller than d to any full trace of N.
- **return** P

At the end of the algorithm, the log traces which are too far (i.e. at distance $> d$) from any run of the model are left unclustered. It is possible then either to increase d in order to cluster those traces, or, if one considers that they are anyway too little related with the model, treat them with another (model agnostic) clustering approach.

5 Implementation and Experiments

We have implemented the theory of this paper in our tool DarkSider, which was initially focused on computing *anti-alignments* [16]. DarkSider is available at `http://www.lsv.ens-cachan.fr/~chatain/darksider`. DarkSider is written in OCaml. For each cluster computed, it constructs the pseudo-Boolean formulas described in Sections 3.2 and calls the pseudo-Boolean solver minisat+ [14]. When an (optimal) solution is found, DarkSider analyses it and displays the corresponding multi-alignment according to the truth values of the variables $\tau_{i,t}$.

To show the capabilities of the tool, we have focused on a synthetic medium-size example (model M1, 40 places and 39 transitions). Figure 3 shows the example. The model contains the typical constructs for a process: sequence, choice, concurrency and loops. It was originally presented in [15], together with a log containing 500 cases of varying sizes. We want to illustrate four different aspects of the contributions of this paper: tool usability, comparison with another clustering approach, combination with other approaches, and resilience to noise.

Tool usability. By assigning values to the few parameters the tool has (that control the length and the distance of the multi-alignments computed), one can obtain a clustering in few minutes. We have run the clustering algorithm with distance[6] 4 and setting 15 as maximal length for a run. As a result, we have obtained 8 clusters, covering 499 traces. The 8 centroids are:

```
c1: A C E G H F D B
c2: A C P W R Y X U Z V T S Q D B
c3: A C AA AF AC AD AE AC AH AJ AI AD AG AE AC
c4: A C AA AC AF AH AJ AI AD AG X2 AB D B
c5: A C E X3 I J K I J K I J K I J
c6: A C AA AF AI AH AJ AC AD AG X2 AB D B
c7: A C P R W Y U Z V T S X Q D B
c8: A C AA AF AJ AI AC AH AD AG AE AC AD AE AC
```

corresponding to the following partitioning of the log in number of traces: (206, 135, 115, 31, 8, 2, 2, 1), respectively. Notice that most of the centroids are full runs of the model (`c1`, `c2`, `c4`, `c6`, `c7`), the others are only prefixes of full runs, truncated to the maximal size (15) that we imposed to centroids. One can see that the aforementioned centroids cover most of the case variants of the model in Figure 3. For instance, centroid `c5` corresponds to the following cluster:

```
t1: J J E I C K I A K I B J F D
t2: A C J I J K I J K F I E D B
t3: A J C K E I J K I J I F D B
t4: C J A E F I J K I D J K I B
t5: A C E J J K I J K I I F D B
t6: A C E I J K I J K B I K I J F D J
t7: A C I K J K E J I I J K I J F D B
t8: A C E I J K I K D I F J K I K I J J J B
```

Overall, the approach can be tuned to provide a coarse-view of the traces (like the one examplified here), or by decreasing the distance, to get a more fine-grained view with more clusters of smaller size. For instance, if we set maximal distance 3 instead of 4, we get 18 clusters.

Comparison with other clustering approach. We compare the results produced by our tool with the technique from [8]. This technique maps every case to a profile vector, and builds a similarity matrix. This matrix is used as input for a *Markov* cluster algorithm, which returns the clustering. In essence, this clustering approach is meant to group together traces with similar labels and behavior. For the same example, this technique provides 18 clusters of sizes ranging from 1 up to 137 traces. A key difference with our approach is the restriction on the traces to be included in a cluster: this approach only obtains clusters with traces having the same length. For instance, the following is an example of one of the clusters obtained by the aforementioned technique:

[6] For efficiency reasons, DarkSideruses currently an ad-hoc distance intermediate between Hamming and Levenshtein

```
t1: A C E J J K I J K I I F D B
t2: C J A E F I J K I D J K I B
t3: A J C K E I J K I J I F D B
t4: A C J I J K I J K F I E D B
t5: J J E I C K I A K I B J F D
```

To force having the same length within a cluster may be the reason why more clusters are obtained in [8]. In case of loops, this can be misleading, as it can be seen in the cluster computed by our tool corresponding to centroid `c5`, where different iterations of the loop (I,J,K) are combined into the same cluster. We believe this is a good feature of our approach, since the core information is the same even if two traces have different number of loop iterations.

Combination with other clustering approach. The theory of this paper can be applied to enrich the information provided by other (model agnostic) trace clustering approaches. Once clusters are produced by any trace clustering technique, obtaining multi-alignments for each one would then provide the model-based centroids as our approach provides. For instance, for the cluster provided above from the approach in [8], our tool produced a multi-alignment having the following model sequence $\langle A, C, E, X3, I, J, K, I, J, X1, F, D, B\rangle$. Also, traces that are left uncovered by our approach (since they are beyond the distance considered) can then be clustered with other approaches that do not consider the model.

Resilience to noise. We have inserted noise into the initial log using the available plugins in the open source tool ProM. The noise insertion was removing and swapping activities in every trace, with a 10% of probability for each one of the noise operations. Accordingly, the clustering approach was computed (with the same parameters) on the same model and the noisy log. Due to the significant insertion of noise, 19 clusters were detected now, and 63 traces out of the 500 became unclustered (compared to the one trace unclustered for the initial log)[7]. In spite of this, some of the centroids were preserved, which give rise to detecting some of the clusters similar to the initial log. For instance the centroid `c1` was again detected, and centroids equivalent (variations of the available concurrency in the model) to `c2`, `c4`, `c6`, `c7` were computed. We performed the same experiment but now with a 20% of probability for each one of the noise operations. The results obtained in terms of centroids and clusters were very similar. Hence, by focusing at the model-based centroid and not at the cluster level, a certain invariance in the results, even in the presence of significant noise, can be obtained. Also, notice that unclustered traces represent model-based outliers that can be then analyzed apart, to drill down the analysis in those cases.

[7] If more flexible distance parameters are applied, a clustering with only 10 traces unclustered can be computed.

6 Conclusion and Perspectives

An important dimension in process mining is to extract from a large log with many traces, high-level information about families of similar traces which may correspond to different executions of the same parts of the model. The techniques in the literature do not consider the process model for solving this task. For the first time, this paper puts the process model as a main actor in trace clustering.

The notion of multi-alignments defined in this paper is a crucial one to facilitate alignment-based clustering. Multi-alignments incorporate the idea of grouping similar traces within the problem of aligning observed behavior with the model. They provide a typical trace which represents as well as possible the behavior of a cluster of similar log traces. These high-level alignments can be viewed as a way to enrich, and at the same time compress, the alignment information. We envision future applications of multi-alignments that will go beyond trace clustering.

Although currently the tool can provide clustering results in few minutes for medium-sized instances, future work will be devoted to improve the efficiency of computing multi-alignments, which is the core part of the clustering algorithm. Our tool computes exact solutions to the optimal multi-alignment problem, which has a cost in terms of efficiency. Finding efficient heuristics for computing reasonable approximations is an interesting perspective, as has been done recently [17]. By considering several traces at the same time, this work opens the door to significant reduction in the overall complexity of aligning an event log and a model, since the number of alignment problems to solve may be considerably lower.

We plan to do an extensive comparison of our technique with respect to other clustering techniques in the literature, over a comprehensive set of benchmarks. We will also show possible combinations of these approaches with ours, with the aim of improving the interpretation of the clustering information obtained.

Acknowledgements

We thank Bart Hompes for facilitating the clustering results of his tool for the example used in the experiments. This work has been partially supported by funds from the Spanish Ministry for Economy and Competitiveness (MINECO), the European Union (FEDER funds) under grant COMMAS (ref. TIN2013-46181-C2-1-R).

References

1. van der Aalst, W.M.P.: Process Mining - Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
2. 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

3. 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, Brisbane, Australia, September 24-28, 2007, Proceedings. (2007) 360–374
4. 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
5. 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, April 30 - May 2, 2009, Sparks, Nevada, USA. (2009) 401–412
6. 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, Ulm, Germany, September 7, 2009. Revised Papers. (2009) 170–181
7. 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
8. 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), Hasselt (Belgium) November 5-6, 2015.
9. 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)
10. Adriansyah, A.: Aligning observed and modeled behavior. PhD thesis, Technische Universiteit Eindhoven (2014)
11. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE **77**(4) (April 1989) 541–574
12. Stewart, I.A.: Reachability in some classes of acyclic Petri nets. Fundam. Inform. **23**(1) (1995) 91–100
13. Cheng, A., Esparza, J., Palsberg, J.: Complexity results for 1-safe nets. Theor. Comput. Sci. **147**(1&2) (1995) 117–136
14. Eén, N., Sörensson, N.: Translating pseudo-boolean constraints into SAT. JSAT **2**(1-4) (2006) 1–26
15. 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), Graz, Austria, December 15-16, 2016. (2016) 50–62
16. Chatain, T., Carmona, J.: Anti-alignments in conformance checking - the dark side of process models. In Kordon, F., Moldt, D., eds.: Application and Theory of Petri Nets and Concurrency - 37th International Conference, PETRI NETS 2016, Toruń, Poland, June 19-24, 2016. Proceedings. Volume 9698 of Lecture Notes in Computer Science., Springer (2016) 240–258
17. Taymouri, F., Carmona, J.: A recursive paradigm for aligning observed behavior of large structured process models. In: Business Process Management - 14th International Conference, BPM 2016, Rio de Janeiro, Brazil, September 18-22, 2016. Proceedings. (2016) 197–214