Encoding Conformance Checking Artefacts in SAT

Mathilde Boltenhagen, Thomas Chatain, Josep Carmona

▶ **To cite this version:**

Mathilde Boltenhagen, Thomas Chatain, Josep Carmona. Encoding Conformance Checking Artefacts in SAT. BPI 2019 - 15th International Workshop on Business Process Intelligence, Sep 2019, Wien, Austria. hal-02419980

HAL Id: hal-02419980
https://inria.hal.science/hal-02419980

Submitted on 19 Dec 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.

Encoding Conformance Checking Artefacts in SAT

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. Conformance checking strongly relies on the computation of artefacts, which enable reasoning on the relation between observed and modeled behavior. This paper shows how important conformance artefacts like alignments, anti-alignments or even multi-alignments, defined over the edit distance, can be computed by encoding the problem as a SAT instance. From a general perspective, the work advocates for a unified family of techniques that can compute conformance artefacts in the same way. The prototype implementation of the techniques presented in this paper show capabilities for dealing with some of the current benchmarks, and potential for the near future when optimizations similar to the ones in the literature are incorporated.

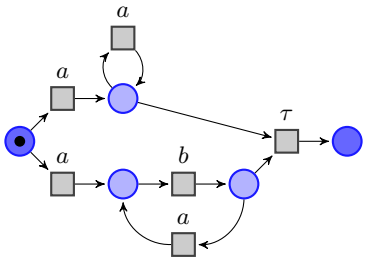
1 Introduction

On its core, conformance checking relies on the computation of artefacts that link observed and modeled behavior, which are used with different purposes: spotting deviations, evaluating quality metrics of a process model, extending a process model with evidence-based information, among others [11].

Conformance checking is expected to be the fastest growing segment within Process Mining in the years to come[3]. Still, the field is facing several challenges. Among them, we highlight two important ones: techniques for a sound replay of event data on top of process models, and the advent of faithful metrics for evaluating process models with respect to observed behavior.

The former challenge is strongly related to the notion of *alignment* artefact: given a trace and a process model, find a run in the process model that is as close as possible (in edit distance terms) to the observed trace. The seminal work in [1] describes an algorithm for computing alignments based on A^*. Alternatives to this algorithm have appeared recently: in [15], the alignment problem is mapped as an *automated planning* instance. Automata-based techniques were proposed in [22,19]. Finally, the work in [5] proposes using binary decision diagrams to alleviate the computation of alignments. In this paper, we provide an alternative to the aforementioned techniques, that is based on encoding the computation of an alignment as a SAT instance. We also show how to encode also in SAT *multi-alignments* [13], that generalize the notion of alignments

[3] https://www.marketsandmarkets.com/Market-Reports/process-analytics-market-254139591.html



(a) Model with two types of runs : $\langle a, a, \ldots, a \rangle$ and $\langle a, b, a, b, \ldots, a, b \rangle$

Log trace : $\langle b, a, b, a, b \rangle$	Hamming distance	Edit distance
run (size = 6) $\langle a, a, a, a, a, a \rangle$	3	6
run (size = 6) $\langle a, b, a, b, a, b \rangle$	6	2

(b) Comparision of Hamming and edit distances

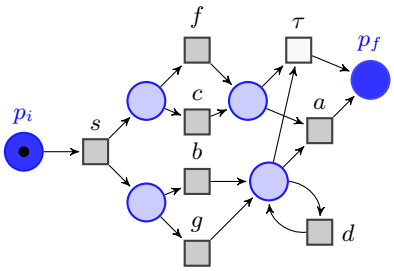
Fig. 1: How Hamming distance penalizes alignment and anti-alignment : Trace $\langle b, a, b, a, b, a \rangle$ is closer to $\langle a, a, a, a, a, a \rangle$ than $\langle a, b, a, b, a, b \rangle$ according to Hamming distance. However there is only a shift of letter a and b between $\langle b, a, b, a, b, a \rangle$ and $\langle a, b, a, b, a, b \rangle$. The model run $\langle a, b, a, b, a, b \rangle$ seems to be a better anti-alignment for trace $\langle b, a, b, a, b, a \rangle$, as the edit distance shows.

so that not one but several traces are considered when computing the closest process model run.

The latter challenge is mainly concerned with the proposal of sound and meaningful metrics for *precision* and *generalization*, nowadays acknowledged as the quality dimensions with less convincing estimations (e.g., [23]). *Anti-alignments*, presented in [12], are an effective conformance artefact to foresee those model runs that deviate most (again, in terms of edit distance) with respect to a trace or a complete log. In [27] it is shown how anti-alignments can be used to provide a more consistent estimation to both precision and generalization. In this paper we provide for the first time an encoding in SAT of anti-alignments that relies on the edit distance between traces, in contrast to the Hamming distance used in [12,27]; to the best of our knowledge this is the first implementation of anti-alignments over the edit distance. The use of Hamming distance for anti-alignment shows important limit highlighted by Fig. 1. In summary, this paper presents SAT as a common means to compute important conformance checking artefacts. The prototype implementation, although not mature and therefore without having any optimization, shows interesting features that can make it a good alternative for the state of the art techniques for the same task in the near future.

The paper is organized as follows: next section provides related techniques for the tasks considered in this paper. Then in Section 2 we provide the background for understanding the paper. Section 3 shows how to encode the computation of a run at a given edit distance of a trace. Then Section 4 shows how to adapt this encoding to particular conformance checking artefacts. Section 5 reports experiments, whilst Section 6 concludes the paper.

Related work. Levenshtein's edit distance is commonly used in Process Mining to define similarities between traces [8,9,24], and to align log traces to model traces [7,1,12]. Works of the literature on alignments use the definition of edit distance more or less



Log traces (some non-fitting):
$\langle s, f, b, a \rangle$
$\langle s, g, c \rangle$
$\langle s, c, b, a \rangle$
$\langle s, g, c, d, d \rangle$
$\langle s, a, a \rangle$

Fig. 2: Example of process model of the behaviors of users rating an app. `s` represents the start activity, `f` and `c` indicate if the user sent a file or wrote a comment. Transitions `g` and `b` separate good and bad marks. Bad ratings get apologies noted by activity `a`. Finally, the `d` loop is enabled when a user donates to the developer of the app.

implicitly [7,16,2]. Alternatively, distance between process models have been characterized in several works [4,17]. The latter uses a dependency edit distance which is computed in a similar way as our approach.

Recent studies focus on SAT implementation of Data Mining algorithm in order to satisfy all the constrains and get optima [20,14]. By introducing a SAT implementation of alignments in this paper, we hope to push a new family of algorithmic methods for conformance checking in the line of [12,6]. However, these works mostly consider Hamming distance between log traces and process models, which is usually considered less appropriate than edit distance (c.f. Fig. 1).

The SAT encoding of the edit distance between words has already been introduced in previous works [3,18]. The work in [3] studies the computation of edit distance for its interest in complexity theory.

2 Preliminaries

We use labeled Petri nets as process models.

Definition 1 (Process Model (Labeled Petri Net) [21]). *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.*

A *marking* is the set of places that contain tokens for a given instant. A transition x can *fire* if all the places before x, noted ${}^\bullet x \stackrel{\text{def}}{=} \{y \in P \mid (y, x) \in F\}$, are marked. When a transition fires, all the tokens in ${}^\bullet x$ are removed and all the places in $x^\bullet \stackrel{\text{def}}{=} \{y \in P \mid (y, x) \in F\}$ become marked. 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 $m[t_1 \dots t_n \rangle m'$.

Definition 2 (Full runs). *A* full run *of a model N is a firing sequence $\langle t_1 \dots t_n \rangle$ of transitions that can transform the initial marking m_0 of N to the final marking m_f of N. We note $Runs(N)$ the full runs of N.*

The run $\langle s, f, b, a \rangle$ is a full run of the model of Fig. 2. Now we formalize logs:

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

2.1 Alignments

Aligning log traces to model traces has been identified as a central problem in Process Mining [1]. The problem is to find a run in the process model that is as close as possible to the observed trace. This closeness is usually defined in terms of the edit distance:

Definition 4 (Edit distance). *The* edit distance *$dist(u, v)$ between two words u and $v \in \Sigma^*$ is the minimal number of edits needed to transform u to v. In our case, edits can be deletions or additions of a letter in words.*

Example 1. Considering words $u = \langle s, g, c \rangle$ and $v = \langle s, b, c, a \rangle$ the number of edits to transform u to v is indeed 3. The letter g has to be removed and the letters b and a inserted. Then the two words are at distance 3.

Formally, we define alignments in a way that not only achieves the optimal edit distance between the log trace and a model trace, but also makes explicit where they match and mismatch.

Definition 5 (Alignment, optimal alignment). *An alignment of a log trace $s = \langle s_1, \dots, s_m \rangle \in L$ to a run $u = \langle u_1, \dots, u_n \rangle$ of process model $N = \langle P, T, F, m_0, m_f, \Sigma, \Lambda \rangle$ is a sequence of* moves *$\langle (s'_1, u'_1), \dots, (s'_p, u'_p) \rangle$ with $p \leq m + n$ such that*

- *each move (s'_i, u'_i) is either:*
  - *(e_i, t_i) with $e_i = \Lambda(t_i)$ for a synchronous move*
  - *$(e_i, \gg)$ for a log move ($\gg$ is a skip symbol which indicates the mismatch), i.e. e_i is deleted in the trace*
  - *$(\gg, t_i)$ for a model move, i.e. $\Lambda(t_i)$ is inserted in the trace;*
- *projection of $\langle s'_1, \dots, s'_p \rangle$ to Σ^* (which drops the occurrences of $\gg$), yields s*
- *projection of $\langle u'_1, \dots, u'_p \rangle$ to T^* (which drops the occurrences of $\gg$), yields u*

An alignment between $u \in Runs(N)$ and s is optimal *if it minimizes the number of occurrences of $\gg$. This minimal number of mismatches corresponds to the edit distance $dist(u, s)$ between u and s.*

Alignments can be represented in a two-row matrix. Next figure shows an alignment between the second log trace ($\langle s, g, c \rangle$) and the run $\langle s, b, c, a \rangle$ of the model of Fig. 2.

trace	s	g	$\gg$	c	$\gg$
run	s	$\gg$	b	c	a

Aligning traces to model often means searching the minimal number of moves, i.e. *the optimal alignment*. Different cost functions are used but the most common one applies the weights 0 for a synchronous move and 1 for a log move or a model move. Next figure shows an alignment for the trace $\langle s, g, c\rangle$ and the run $\langle s, g, c, \tau\rangle$ of the model of Fig. 2. Since τ moves inccur into no cost, the alignment has cost 0, and hence it is optimal.

trace	s	g	c	$\gg$
run	s	g	c	τ

3 SAT Encoding of the Edit Distance

We first introduce our SAT encoding of the edit distance between two words, which will serve as a building block for alignment, multi-alignment and anti-alignment.

The Boolean satisfiability (or SAT) problem, is the problem of determining, for a given Boolean formula, if there exists a combination of assignments to the variables that satisfies it. In the case of alignment, a SAT formula would encode the following question: *Does it exist an alignment between the trace σ and the model N for a cost d ?* In other words, we are looking for a formulas that encodes *the edit distance d between a trace and a run of the model*. Our encoding is based on the same relations that are used by the classical dynamic programming recursive algorithm for computing the edit distance between two words $u = \langle u_1, \ldots, u_n\rangle$ and $v = \langle v_1, \ldots, v_m\rangle$:

$$\begin{cases} dist(\langle u_1, \ldots, u_i\rangle, \epsilon) = i \\ dist(\epsilon, \langle v_1, \ldots, v_j\rangle) = j \\ dist(\langle u_1, \ldots, u_{i+1}\rangle, \langle v_1, \ldots, v_{j+1}\rangle) = \\ \quad \begin{cases} dist(\langle u_1, \ldots, u_i\rangle, \langle v_1, \ldots, v_j\rangle) & \text{if } u_{i+1} = v_{j+1} \\ 1 + \min(dist(\langle u_1, \ldots, u_{i+1}\rangle, \langle v_1, \ldots, v_j\rangle), & \\ \qquad dist(\langle u_1, \ldots, u_i\rangle, \langle v_1, \ldots, v_{j+1}\rangle)) & \text{if } u_{i+1} \neq v_{j+1} \end{cases} \end{cases}$$

We encode this computation in a SAT formula ϕ over variables $\boldsymbol{\delta_{i,j,d}}$, for $i = 0, \ldots, n$, $j = 0, \ldots, m$ and $d = 0, \ldots, n + m$. The formula ϕ will have exactly one solution, in which each variable $\boldsymbol{\delta_{i,j,d}}$ is `true` iff $dist(\langle u_1 \ldots u_i\rangle, \langle v_1 \ldots v_j\rangle) \geq d$.

In order to test equality between the u_i and v_j, we use variables $\boldsymbol{\lambda_{i,a}}$ and $\boldsymbol{\lambda'_{j,a}}$, for $i = 0, \ldots, n$, $j = 0, \ldots, m$ and $a \in \Sigma$, and we set their value such that $\lambda_{i,a}$ is `true` iff $u_i = a$, and $\lambda'_{j,a}$ is `true` iff $v_j = a$. Hence, the test $u_{i+1} = v_{j+1}$ becomes in our formulas: $\bigvee_{a\in\Sigma}(\lambda_{i+1,a} \wedge \lambda'_{j+1,a})$. For readability of the formulas, we refer to this coding by $[u_{i+1} = v_{j+1}]$. We also write similarly $[u_{i+1} \neq v_{j+1}]$.

In the following, we describe the different clauses of the formula ϕ of our SAT encoding of the edit distance.

$$\delta_{0,0,0} \quad \wedge \quad \bigwedge_{d>0} \neg\delta_{0,0,d} \tag{1}$$

$$\bigwedge_d \bigwedge_{i=0}^{n} \quad (\delta_{i+1,0,d+1} \Leftrightarrow \delta_{i,0,d}) \tag{2}$$

$$\bigwedge_d \bigwedge_{j=0}^{n} \quad (\delta_{0,j+1,d+1} \Leftrightarrow \delta_{0,j,d}) \tag{3}$$

$$\bigwedge_d \bigwedge_{i=0}^{n} \bigwedge_{j=0}^{n} \quad [u_{i+1} = v_{j+1}] \Rightarrow (\delta_{i+1,j+1,d} \Leftrightarrow \delta_{i,j,d}) \tag{4}$$

$$\bigwedge_d \bigwedge_{i=0}^{n} \bigwedge_{j=0}^{n} \quad [u_{i+1} \neq v_{j+1}] \Rightarrow (\delta_{i+1,j+1,d+1} \Leftrightarrow (\delta_{i+1,j,d} \wedge \delta_{i,j+1,d})) \tag{5}$$

Example 2. At instants $i = 1$ and $j = 1$ of words $u = \langle s, g, c\rangle$ and $v = \langle s, b, c, a\rangle$, the letters are the same, then, by (4), the distance is only higher or equal to 0 : $(u_1 = v_1) \Rightarrow (\delta_{1,1,0} \Leftrightarrow \delta_{0,0,0})$.

However at instants $i = 2$ and $j = 2$, the letters u_2 and v_2 are different. A step before, $\delta_{1,2,1}$ and $\delta_{2,1,1}$ are `true` because of the length of the subwords. Then, by (5), the distance at instants $i = 2$ and $j = 2$ is higher or equal to 2 : $\delta_{2,2,2}$. The result is understandable because the edit distance costs the deletion of g and the addition of b to transform u to v.

4 SAT Encoding of Conformance Checking Artefacts

The distance between log traces and a process model is not only a distance between words, since process model describe a (possibly infinite) language. In this part, we recall the SAT encoding of full runs of Petri nets [12] and combine the implementation with the edit distance.

SAT implementation of process models. For a Petri net $N = \langle P, T, F, m_0, m_f, \Sigma, \Lambda\rangle$ and n the size of the full runs, the variables $\boldsymbol{m}_{i,p}$, with $i \in \{0..n\}$ and $p \in P$, represent the marking at instant i. The variables $\boldsymbol{\tau}_{i,a}$ encode a firing transition $t \in T$ labelled by $a \in \Sigma$ at instant $i \in \{0..n\}$[4]. The following constraints encode the semantics of the Petri net.

- Initial marking:
$$(\bigwedge_{p\in m_0} m_{0,p}) \wedge (\bigwedge_{p\in P\setminus m_0} \neg m_{0,p}) \tag{6}$$
- Final marking:
$$(\bigwedge_{p\in m_f} M_{n,p}) \wedge (\bigwedge_{p\in P\setminus m_f} \neg m_{n,p}) \tag{7}$$
- One and only one t_i for each i:
$$\bigwedge_{i=1}^{n} \bigvee_{a\in\Sigma} (\tau_{i,a} \wedge \bigwedge_{a'\in\Sigma\setminus t} \neg\tau_{i,a'}) \tag{8}$$

[4] For the sake of simplicity of the encoding, we are abusing a bit the notation, i.e., assuming that labels identify transitions. This can be generalized easily for the general case when severals transitions exist for the same label.

– The transitions are enabled when they fire:

$$\bigwedge_{i=1}^{n} \bigwedge_{a\in\Sigma} (\tau_{i,a} \implies \bigwedge_{p\in {}^\bullet t} m_{i-1,p}) \tag{9}$$

– Token game (for safe Petri nets):

$$\bigwedge_{i=1}^{n} \bigwedge_{a\in\Sigma} \bigwedge_{p\in t^\bullet, \Lambda(t)=a} (\tau_{i,a} \implies m_{i,p}) \tag{10}$$

$$\bigwedge_{i=1}^{n} \bigwedge_{a\in\Sigma} \bigwedge_{p\in {}^\bullet t\setminus t^\bullet, \Lambda(t)=a} (\tau_{i,a} \implies \neg m_{i,p}) \tag{11}$$

$$\bigwedge_{i=1}^{n} \bigwedge_{a\in\Sigma} \bigwedge_{p\in P, p\notin {}^\bullet t, p\notin t^\bullet, \Lambda(t)=a} (\tau_{i,a} \implies (m_{i,p} \iff m_{i-1,p})) \tag{12}$$

Process models are now implemented in a CNF formula and can be combined with log traces for alignments.

4.1 SAT Edit Distance for Alignments

Log traces are sequences of activities that can be considered as words and implemented as presented in Section 3. SAT encoding of process models has been recalled in previous section. All the above clauses are considered in the SAT implementation of alignments. A last series of constraints is needed to be appended, to relate the fired transitions, represented by the $\boldsymbol{\tau}_{i,a}$, with the actions in the corresponding model trace, represented by variables $\boldsymbol{\lambda}_{i,a}$ from the encoding of Section 3:

$$\bigwedge_{i=1}^{n} \bigvee_{a\in\Sigma} (\lambda_{i,a} \iff \tau_{i,a}) \tag{13}$$

Example 3. All the full runs of the process model of Fig. 2 contain a s at the first instant. So the variable $\tau_{1,s}$ is `true`. If the log trace is $\sigma = \langle s, f, g\rangle$ then, $\lambda_{1,s}$ is `true` which implies $\delta_{1,1,0}$ by (4).

Minimization of the edit distance. The conjunction of the previous clauses for the full runs of the model and the their edit distance to a given log trace σ, gives a formula which has one solution per full run of the model. With each solution, the values of the $\delta_{n,m,d}$ determine the edit distance between the corresponding model trace and σ. Our goal for optimal alignments is to minimize this distance, which corresponds to the number of variables assigned to `true` among the $\delta_{n,m,d}$. Pseudo-Boolean solvers like MINISAT+ deal with minimization objectives under the form of a weighted sum of variables; in our case: $\sum_d 1 \times \delta_{n,m,d}$.

How to deal with runs of different length. In order to consider different sizes of traces and different sizes of runs, we added a loop on a *wait* activity on the final marking of the model. The SAT encoding of the edit distance is adjusted so that skipping a *wait* activity does not increment the distance between words.

Example 4. Fig. 3 shows optimal alignments of every trace of the log of Fig. 2. The deviating trace $\langle s, a, a\rangle$ is then highlighted by the distance to its alignment.

Trace	Alignment	Distance
$\langle s, f, b, a\rangle$	$\langle s, f, b, a\rangle$	0
$\langle s, g, c\rangle$	$\langle s, g, c, \tau\rangle$	0
$\langle s, c, b, a\rangle$	$\langle s, c, b, a\rangle$	0
$\langle s, g, c, d, d\rangle$	$\langle s, g, c, d, \tau, d\rangle$	0
$\langle s, a, a\rangle$	$\langle s, b, c, a\rangle$	3

Fig. 3: Alignment of each trace and the model of Fig. 2

4.2 SAT Implementation for Multi-alignments

Multi-alignments were introduced in [13] as a generalization of alignments. Multi-alignments were used to define a model-based trace clustering method. Instead of aligning a trace to a run of a process model, they align a set of log traces (typically from the same cluster) to a common run of the model.

Definition 6 (Multi-alignment). *Given a finite collection C of log traces and a model N, an (optimal)* multi-alignment *of C to N is a full run $u \in Runs(N)$ which minimizes the sum $\sum_{\sigma \in C} dist(\sigma, u)$.*

The SAT implementation of multi-alignment requires us to duplicate the variables $\lambda^{\sigma}_{i,a}$ that represent actions in the log traces $\sigma \in L$ and the variables $\delta^{\sigma}_{i,j,d}$ that measure the edit distance to the model trace. Similarly to Section 4.1, the optimal multi-alignment is found by minimizing the number of variables assigned to `true` in the following objective: $\sum_d \sum_{\sigma \in L} 1 \times \delta^{\sigma}_{n,|\sigma|,d}$.

Example 5. We computed the multi-alignment of the model and the full log of Fig. 2. The optimal multi-alignment is the full run $\langle s, f, b, a\rangle$ which is at distance $d \leq 3$ to all the log traces.

4.3 SAT Implementation for Anti-alignments

Anti-alignment was introduced in [12]. Contrary to multi-alignments, the aim of anti-alignments is to get, for a given log, the run of a model which differs as much as possible to all the traces in the log. The notion of anti-alignment is used in some quality metrics like precision and generalization [26].

Definition 7 (Anti-alignment). *Given a finite collection L of log traces and a model N, an* anti-alignment *is a run $u \in Runs(N)$ which maximizes its distance $\sum_{\sigma \in L} dist(\sigma, u)$ to the log.*

The encoding is then very similar to the multi-alignment version. Instead of minimizing the distance to the set of log traces, we maximize it using the opposite minimization objective: $\sum_d \sum_{\sigma \in L} -1 \times \delta^{\sigma}_{n,|\sigma|,d}$[5].

[5] Only the total sum of δ are minimized/maximized in our tool

Example 6. We computed the anti-alignment of model and the set of log traces of Fig. 2 . Limited by a maximum size of run to 8, the optimal anti-alignment found by our tool is $\langle s, b, d, f, d, d, d, \tau \rangle$. The minimal distance between each trace and this full run is 9. We compared our result with the module *Anti-Alignment* of ProM [6], that computes anti-alignment over Hamming distance. For the same size of run, the algorithm returned the sequence $\langle s, b, d, d, d, c, a, d \rangle$ that is indeed linearly far from the log traces. However as either letters c or a are present in every trace, the run looks more similar to the log than the one found with edit distance.

5 Experiments

The construction of SAT formulas for alignments, multi-alignments and anti-alignments is implemented in Ocaml in our tool DARKSIDER available on github[7]. The software invokes a SAT solver, by default MINISAT+. Examples of the previous section have been fully computed by the SAT formulas to get optimal solutions. Since the approach is heavily influenced by the size of the formulas constructed (which is large even for small models), in this section we focus on heuristics to simplify the formulas, at the expense of sacrifying optimality eventually. Those are preliminary results oriented towards illustrating how to instruct the current tool.

A first heuristic is to remove $\delta_{i,j,d}$ variables, for d large . Limiting the maximal number of edits may not change the result when the traces are close to the model. A second approximation is to get optimal prefix alignment by limiting the size of the run.

Tab. 1 (a)-(c) summarizes the experiments. Notice that while for alignments we show average numbers over one hundred traces, for multi- and anti-alingments (where only a run is computed for the whole log) total numbers are provided. The first two columns (Model and $|L|$) describe the model and the log size, respectively. Column "Size of run" shows the maximal size allowed for the run in the model (which will be an alignment in (a), a multi-alignment in (b), and an anti-alignment in (c)). Sometimes the length is limited when PRE is specified, as explained above. Then the fourth column reports the maximal number of edits allowed, sometimes with a bound as explained above. When LIM is indicated, the distance between the model and the trace is larger than what was tested. The last three columns show the time to construct the formula and the total execution time for our approach, and the time needed in ProM. Notice that since we are providing results for edit-distance based conformance artefacts, only ProM results for alignments are shown.

The goal of this paper was to demonstrate the interest of our approach based on a SAT encoding of Petri net executions and edit distance between traces. For alignments, indeed, the current SAT encoding shows bad execution times compared to the optimized algorithms implemented in ProM. But we obtain, for the first time, an implementation of the anti-alignment and multi-alignment artefacts defined over the edit distance. We believe that there is a lot of space for improvement, for instance by optimizing the encoding or by using heuristics to very efficiently find approximations of the results.

[6] Anti-alignment Precision/Generalization of package AntiAlignments of ProM software version 6.8, http://www.promtools.org/

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

Model			$\|L\|$	Size of run	Maximal number of edits	Formula construction time (sec)	Total execution time (sec)	ProM execution time (sec)
Reference	$\|T\|$	$\|P\|$						
Fig. 2	8	7	100	7	5	0.239	0.349	0.002
M8 of [25]	15	17	100	PRE: 20	LIM:10	10.139	15.530	0.001
M1 of [25]	40	39	100	PRE: 7	LIM:10	4.924	7.16	0.005
Loan [10]	15	16	100	PRE: 19	LIM: 10	14.047	20.915	0.002

(a) Alignments (showing averages).

Fig. 2	8	7	10	8	7	10.101	15.362	/
			100	8	7	99.602	200.569	
M8 of [25]	15	17	10	18	LIM:6	252.471	414.174	/
			100	PRE:15	LIM:6	516.391	741.162	
M1 of [25]	40	39	10	PRE: 13	LIM:10	115.706	172.500	/
			100	PRE: 13	LIM: 5	681.95	1066.94	
Loan [10]	15	16	10	PRE: 19	15	252.572	373.683	/
			100	PRE: 9	LIM:10	359.982	508.542	

(b) Multi-alignments.

Fig. 2	8	7	10	8	LIM: 10	13.802	21.502	/
			100	8	LIM: 10	137.213	243.842	
M8 of [25]	15	17	10	18	LIM:10	103.812	148.271	/
			100	PRE: 10	LIM: 10	343.529	496.733	
M1 of [25]	40	39	10	39	LIM:10	1337.806	2069.505	/
			100	PRE:13	LIM:5	680.556	995.361	
Loan [10]	15	16	10	PRE: 19	LIM: 10	140.840	203.257	/
			100	PRE:19	LIM: 10	1526.048	2185.785	

(c) Anti-alignments.

Table 1: Experimental results for the computation of optimum and approximations of alignments and anti-alignments with our tool DARKSIDER, obtained on a virtual machine with CPU Intel Xeon 2.67GHz and 50GB RAM.

6 Conclusion

This paper has shown a unified approach to compute important conformance artefacts over SAT. Thanks to its high versatility, the encoding as SAT formula allows one to compute exact solutions to various problems (here alignments, anti-alignments and multi-alignments) under various optimality criteria. In particular, we show for the first time how anti-alignments can be computed for the edit distance by formulating the problem as a SAT instance. Although technically sound, the encodings proposed in this paper suffer from the explosion of the SAT formulas created. As main research direction, we are working into finding better encodings that can alleviate significantly the size of such formulae, but also incorporating other optimizations and heuristics (possibly inspired from optimization techniques used in automated planning) that can make the approach more efficient in practice.

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. A. Adriansyah. *Aligning observed and modeled behavior*. PhD thesis, Department of Mathematics and Computer Science, 2014.
2. Arya Adriansyah, Jorge Munoz-Gama, Josep Carmona, Boudewijn F van Dongen, and Wil MP van der Aalst. Alignment based precision checking. In *International Conference on Business Process Management*, pages 137–149. Springer, 2012.
3. Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless seth is false). In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 51–58. ACM, 2015.
4. Joonsoo Bae, Ling Liu, James Caverlee, Liang-Jie Zhang, and Hyerim Bae. Development of distance measures for process mining, discovery and integration. *International Journal of Web Services Research (IJWSR)*, 4(4):1–17, 2007.
5. Vincent Bloemen, Jaco van de Pol, and Wil M. P. van der Aalst. Symbolically aligning observed and modelled behaviour. In *18th International Conference on Application of Concurrency to System Design, ACSD, Bratislava, Slovakia, June 25-29*, pages 50–59, 2018.
6. Mathilde Boltenhagen, Thomas Chatain, and Josep Carmona. Generalized alignment-based trace clustering of process behavior. In *Proceedings of the 40th International Conference on Applications and Theory of Petri Nets (ICATPN'19)*, number 11522 in Lecture Notes in Computer Science. Springer, 2019.
7. RP Jagadeesh Chandra Bose and Wil van der Aalst. Trace alignment in process mining: opportunities for process diagnostics. In *International Conference on Business Process Management*, pages 227–242. Springer, 2010.
8. RP Jagadeesh Chandra Bose and Wil MP Van der Aalst. Abstractions in process mining: A taxonomy of patterns. In *International Conference on Business Process Management*, pages 159–175. Springer, 2009.
9. RP Jagadeesh Chandra Bose and Wil MP Van der Aalst. Context aware trace clustering: Towards improving process mining results. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, pages 401–412. SIAM, 2009.
10. J.C.A.M. Buijs. Loan application example. 4TU. Centre for Research Data. Dataset. doi.org/10.4121, 2013.

11. Josep Carmona, Boudewijn F. van Dongen, Andreas Solti, and Matthias Weidlich. *Conformance Checking - Relating Processes and Models*. Springer, 2018.
12. Thomas Chatain and Josep Carmona. Anti-alignments in conformance checking–the dark side of process models. In *International Conference on Application and Theory of Petri Nets and Concurrency*, pages 240–258. Springer, 2016.
13. Thomas Chatain, Josep Carmona, and Boudewijn Van Dongen. Alignment-based trace clustering. In *International Conference on Conceptual Modeling*, pages 295–308. Springer, 2017.
14. Ian Davidson, SS Ravi, and Leonid Shamis. A sat-based framework for efficient constrained clustering. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 94–105. SIAM, 2010.
15. Massimiliano de Leoni and Andrea Marrella. Aligning real process executions and prescriptive process models through automated planning. *Expert Syst. Appl.*, 82:162–183, 2017.
16. Massimiliano De Leoni and Wil MP van der Aalst. Data-aware process mining: discovering decisions in processes using alignments. In *Proceedings of the 28th annual ACM symposium on applied computing*, pages 1454–1461. ACM, 2013.
17. Remco Dijkman, Marlon Dumas, Luciano Garcia-Banuelos, and Reina Kaarik. Aligning business process models. In *2009 IEEE International Enterprise Distributed Object Computing Conference*, pages 45–53. IEEE, 2009.
18. Alex Groce, Sagar Chaki, Daniel Kroening, and Ofer Strichman. Error explanation with distance metrics. *International Journal on Software Tools for Technology Transfer*, 8(3):229–247, 2006.
19. Sander J. J. Leemans, Dirk Fahland, and Wil M. P. van der Aalst. Scalable process discovery and conformance checking. *Software and System Modeling*, 17(2):599–631, 2018.
20. Jean-Philippe Métivier, Patrice Boizumault, Bruno Crémilleux, Mehdi Khiari, and Samir Loudni. Constrained clustering using sat. In *International Symposium on Intelligent Data Analysis*, pages 207–218. Springer, 2012.
21. T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–574, April 1989.
22. Daniel Reißner, Raffaele Conforti, Marlon Dumas, Marcello La Rosa, and Abel Armas-Cervantes. Scalable conformance checking of business processes. In *OTM CoopIS, , Rhodes, Greece*, pages 607–627, 2017.
23. Niek Tax, Xixi Lu, Natalia Sidorova, Dirk Fahland, and Wil M. P. van der Aalst. The imprecisions of precision measures in process mining. *Inf. Process. Lett.*, 135:1–8, 2018.
24. Niek Tax, Natalia Sidorova, Reinder Haakma, and Wil MP van der Aalst. Event abstraction for process mining using supervised learning techniques. In *Proceedings of SAI Intelligent Systems Conference*, pages 251–269. Springer, 2016.
25. Farbod Taymouri and Josep Carmona. 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.*, pages 50–62, 2016.
26. Boudewijn F. van Dongen, Josep Carmona, and Thomas Chatain. A unified approach for measuring precision and generalization based on anti-alignments. In *Business Process Management - 14th International Conference, BPM 2016, Rio de Janeiro, Brazil, September 18-22, 2016. Proceedings*, pages 39–56, 2016.
27. Boudewijn F. van Dongen, Josep Carmona, Thomas Chatain, and Farbod Taymouri. Aligning modeled and observed behavior: A compromise between computation complexity and quality. In *Advanced Information Systems Engineering - 29th International Conference, CAiSE 2017, Essen, Germany, June 12-16, 2017, Proceedings*, pages 94–109, 2017.