Partial MaxSAT Computation of Conformance Checking Artefacts

Jesus Ojeda
Logic & Optimization Group (LOG)
University of Lleida
Lleida, Spain
jesus.ojedacontreras@udl.cat

***Abstract*—To reason about observed behaviour of processes and their models, conformance checking techniques are rooted in the computation of artefacts. Related artefacts like alignments, multi-alignments and anti-alignments are defined over a distance function, most commonly Hamming or Levenshtein distances. In this paper we provide a new Partial MaxSAT encoding of these artefacts based on the Levenshtein distance and compare with their current state-of-the-art SAT encodings. We show a reduction in the resulting formula size for our proposed encoding, while also obtaining good performance results on the computation of the artefacts.**

***Index Terms*—Alignment, Conformance Checking, MaxSAT**

I. Introduction

Process models play an important aspect in describing, analysing and monitoring the execution of the particular processes executed in the day-to-day operations of organizations. These process models may refer to any representation of a process, although the most common idea is that of workflow process models, one that captures the actions (and their order of execution) involved in the process, and can either be created by a human or by algorithmic means. This concept gains strength in the current world of digital transformation where organizations want to take advantage of all data generated by their own operations and the interactions with their clients.

Then, conformance checking, a particular sub-field within process mining, aims to analyse the relation of observed behaviour as recorded by an event log, with the provided process model or, in other words, how well the model describes the observed reality. This analysis is rooted in the computation of conformance artefacts which may serve different purposes, from detecting deviations to computing quality metrics for the optimization of organization processes, among others [1].

Two main dimensions to assess the relation of process models and event logs are commonly used in conformance checking: fitness (how well can the model reproduce the log?), and precision (is all modelled behaviour present in the log?). For both of them, the concept of *alignment* [2] plays an important role. Given a process model and a trace in the event log, an alignment provides the run in the model that mostly resembles that of the trace. *Multi-alignments* [3] expand the concept so several traces are considered at the same time when computing a run in the model, the closest run to all traces simultaneously.

In contrast, *Anti-alignments* [4] provide a run of the model that mostly deviates, i.e., is most different, from the traces present in the log. Anti-alignments can be useful to detect exotic model runs in contexts where homogeneous behaviour is desired, like in healthcare. They are also related to the concept of completeness of the log and precision, previously mentioned.

Both alignments (single or multi-) and anti-alignments base the similarity or deviation computation in terms of an edit distance. Their implementation can be approached from a satisfiability standpoint with some recent works in this line [5]–[7]. Our contribution in this regard is a new encoding into Partial MaxSAT, an optimization version and generalization of the satisfiability problem. This encoding greatly reduces the size of the generated MaxSAT formulas for full traces in comparison to the state-of-the-art encoding while also outperforming the time required to solve the problem instance.

This paper is structured as follows. First, Section II presents the related work, then Section III introduces some definitions. Section IV presents our Partial MaxSAT encoding of conformance artefacts. Section V provides the experimental results and the paper is concluded in Section VI with some future work.

II. Related Work

Alignments were defined in [2], where the A* algorithm is used to search over the synchronous product of model and trace [8]. There are, however, alternative works based on other techniques for the computation of alignments.

For example, [9] tackles the alignment computation by representing the conformance checking problem into a planning problem; the authors translate the alignment search into the Planning Domain Definition Language (PDDL) and use a state-of-the-art planner. Binary Decision Diagrams (BDD) are used in [10], here the model and log are represented within the same Petri net, splitting its transitions into moves with and without cost; a shortest path in this Petri net then reports the alignment.

There are also techniques based on finite automata. In [11], the authors create a reachability graph from the process model and a minimal Deterministic Acyclic Finite Automaton from the event log; with them, an error-correcting synchronized product automaton is computed that enables to enumerate a

2021 3rd International Conference on Process Mining (ICPM 2021), available at https://doi.org/10.1109/ICPM53251.2021

set of optimal trace alignments. Alternatively, [12] projects both model and log into Deterministic Finite Automata and compare them using their conjunction to compute precision and fitness.

Decomposition techniques also exist. The works from [13] and [14] partition larger process models and event logs into smaller parts that can be analyzed, i.e., search for an alignment, independently.

A different approach based on Integer Linear Programming (ILP) is presented in [15]. The alignments computed are *approximate alignments*, in which the granularity of the moves is user-defined and deviations can be explained by non-unitary sets of activities instead of single pairs. The same authors present in a more recent publication [16] a local search method that improves an alignment from a replay technique.

Finally, there are recent publications that use Satisfiability (SAT) in Process Mining and Conformance Checking [4]–[7], [17]. As with [9], by reframing (encoding) the alignment computation problem in terms of SAT, the problem remains agnostic of the solver used, taking advantage of state-of-the-art SAT and MaxSAT solver technology as it improves over time. In particular, [4] introduces the computation of anti-alignments with a SAT formulation and a Hamming distance. Its same tool is used in [17] to implement trace clustering. The encoding is evolved in [5] and [6] to provide a family that can solve for alignments, multi- and anti-alignments, and then again reused in [7] as an improved version of [4].

In this paper we also use SAT, in particular Partial MaxSAT, to compute alignments. We adapt the same Petri net encoding from the previous SAT-based works but create a new encoding for the edit distance in order to reduce the memory footprint requirements. Our encoding is also easily extended to multi- and anti-alignments.

III. Preliminaries

In this Section, we introduce some definitions that establish the base upon which the rest of the paper is built. Definitions 1 to 5 formally define the concepts of Petri nets, runs and logs used in Process Mining, then Definitions 6 to 14 define the satisfiability concepts and more specifically Partial MaxSAT and pseudo-Boolean constraints we use in our encoding.

A. Petri net, runs and log definitions

Definition 1. *A Petri net is a tuple $N = (P, T, F, W, m_0)$, 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, $W : F \to \{w_1, w_2, \dots\}$ is a weight function and m_0 is the initial marking.*

A marking assigns to each place p a non-negative number of tokens k. A transition t is enabled (can be fired) if all places p before t (the pre-set) are marked: ${}^\bullet t = \{p \in P : (p, t) \in F\}$. A firing of an enabled transition t, removes $w(p, t)$ tokens from each p in ${}^\bullet t$ and adds $w(t, p)$ tokens to each output place p of t (the post-set), defined as $t^\bullet = \{p \in P : (t, p) \in F\}$. 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'.*

Definition 2. *A place p in a Petri net is k-bounded if it does not contain more that k tokens in all reachable markings. The Petri net is k-bounded or* safe *if all places p are k-bounded.*

To use Petri nets as process models we are interested in using 1-bounded or safe Petri nets, while also restricting the arc weights to be unitary from the common Petri net definition (Definition 1). Additionally, because process mining considers complete traces and not a prefix-closed language, we use an extended Petri net that considers a final marking and a set of labels. See Figure 1 for an example process model Petri net.

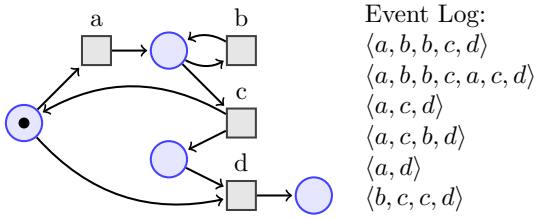

Fig. 1. Example of a process model Petri net and a possible event log (not all traces fitting).

Definition 3. *A process model Petri net system is a tuple $M = (P, T, F, m_0, m_f, \Sigma, A)$, where P, T, F, m_0 define a safe Petri net with unitary weights, m_f is the final marking, Σ is an alphabet of actions and $A : T \to \Sigma$ labels every transition by an action.*

Definition 4. *A full run $\mathcal{V}$ of a model M is a transition firing sequence $\langle t_1 \dots t_n \rangle$ that transforms the initial marking m_0 into the final marking m_f of M. The set of all possible full runs is denoted as $\mathcal{R}_M$.*

Notice that it is assumed that m_f is reachable from m_0. Also, applying function A to each t in firing sequence $\langle t_1 \dots t_n \rangle$ gives a word built from alphabet Σ.

Definition 5. *A log L over an alphabet Σ is a finite set of finite words built from that alphabet, i.e., $L \subseteq \Sigma^*$. An element $\Lambda \in L$ is called a log trace.*

B. MaxSAT definitions

Definition 6. *A literal is a propositional variable x or a negated propositional variable $\neg x$. A clause is a disjunction of literals. A formula or instance ϕ in Conjunctive Normal Form (CNF) is a conjunction of clauses.*

Definition 7. *A weighted clause is a pair (c, w), where c is a clause and w, its weight, is a natural number or infinity. A clause is hard if its weight is infinity (or no weight is given); otherwise, it is soft. A Weighted Partial MaxSAT instance is a multiset of weighted clauses.*

Definition 8. *A truth assignment for an instance ϕ is a mapping that assigns to each propositional variable in ϕ either 0 (False) or 1 (True). A truth assignment is* partial *if the mapping is not defined for all the propositional variables in ϕ.*

Definition 9. *A truth assignment I satisfies a literal x ($\neg x$) if I maps x to 1 (0); otherwise, it is falsified. A truth assignment I satisfies a clause if I satisfies at least one of its literals; otherwise, it is violated or falsified. The cost of a clause (c, w) under I is 0 if I satisfies the clause; otherwise, it is w. Given a partial truth assignment I, a literal or a clause is undefined if it is neither satisfied nor falsified.*

Definition 10. *The cost of a formula ϕ under a truth assignment I, denoted by $cost(I, \phi)$, is the aggregated cost of all its clauses under I.*

Definition 11. *The Weighted Partial MaxSAT (WPMaxSAT) problem for an instance ϕ is to find an assignment in which the sum of weights of the falsified soft clauses is minimal (referred to as the optimal cost of ϕ) and all the hard clauses are satisfied. The Partial MaxSAT problem is the WPMaxSAT problem when all the soft clauses have the same weight. The MaxSAT problem is the Partial MaxSAT problem when there are no hard clauses. The SAT problem is the Partial MaxSAT problem when there are no soft clauses.*

Definition 12. *An instance of Weighted Partial MaxSAT, or any of its variants, is unsatisfiable if its optimal cost is ∞. A SAT instance ϕ is satisfiable if there is a truth assignment I, called model, such that $cost(I, \phi) = 0$.*

Definition 13. *An* At-Most-One *(AMO) constraint is a cardinality constraint of the form $\sum_{i=1}^{n} l_i \leq 1$, where l_i are literals.*

Definition 14. *An* Exactly-One *(EO) constraint is a cardinality constraint of the form $\sum_{i=1}^{n} l_i = 1$, where l_i are literals.*

We assume that cardinality constraints are translated into CNF through the regular encoding [18], [19], and the typical transformations [20] of propositional formulas into CNF are implicitly applied.

IV. Partial MaxSAT encodings

Here we present our Partial MaxSAT encodings for optimal conformance checking artefacts, in particular, alignments, multi-alignments and anti-alignments. First we will provide the SAT encoding of the process model Petri net and then will augment it for each artefact in subsequent sections.

For the Petri net, we use a similar encoding to [3]–[6], that we recall here.

For a process model Petri net M as in Definition 3, and N being the size of the runs, we consider the following boolean variables (where $[N] = \{1, \ldots, N\}$):

- $m_{i,p}$ for $i \in [N]$ and $p \in P$ indicates if place p at instant i is marked (has a token).
- $\nu_{i,t}$ for $i \in [N]$ and $t \in T$ means that transition t is fired at instant i.

Then, the Petri net is encoded with the conjunction of the following formulae:

$$\left(\bigwedge_{p \in m_0} m_{0,p}\right) \wedge \left(\bigwedge_{p \in P \setminus m_0} \neg m_{0,p}\right) \tag{1}$$

$$\left(\bigwedge_{p \in m_f} m_{n,p}\right) \wedge \left(\bigwedge_{p \in P \setminus m_f} \neg m_{n,p}\right) \tag{2}$$

$$\bigwedge_{i \in [N]} \sum_{t \in T} \nu_{i,t} = 1 \tag{3}$$

$$\bigwedge_{i \in [N]} \bigwedge_{t \in T} \left(\nu_{i,t} \rightarrow \bigwedge_{p \in {}^\bullet t} m_{i-1,p}\right) \tag{4}$$

$$\bigwedge_{i \in [N]} \bigwedge_{t \in T} \bigwedge_{p \in t^\bullet} (\nu_{i,t} \rightarrow m_{i,p}) \tag{5}$$

$$\bigwedge_{i \in [N]} \bigwedge_{t \in T} \bigwedge_{p \in {}^\bullet t \setminus t^\bullet} (\nu_{i,t} \rightarrow \neg m_{i,p}) \tag{6}$$

$$\bigwedge_{i \in [N]} \bigwedge_{t \in T} \bigwedge_{p \in P, p \notin {}^\bullet t, p \notin t^\bullet} (\nu_{i,t} \rightarrow (m_{i,p} \leftrightarrow m_{i-1,p})) \tag{7}$$

Constraints 1 and 2 enforce initial and final markings. Constraint 3 limits to only one (EO) transition t to be fired per instant i. Constraints 4–7 implement the token restrictions for the activation of transitions. In particular, Constraint 4 requires the pre-set of t to be marked at the previous instant, for t to fire at instant i. If a transition t fires at instant i, then tokens for its post-set have to be added (Constraint 5) and tokens from its pre-set and not in the post-set have to be removed (Constraint 6). Finally, with Constraint 7 all marked places that are unaffected by the fired transition t at instant i have to maintain their tokens.

A. Alignments

One central problem of Process Mining is the alignment of log traces to model runs [2], which can be summarized as finding a run in the process model that is as close as possible to the observed trace. As both log traces and model runs can be described as finite words over an alphabet Σ, this closeness is computed in terms of some distance over the two sequences.

Common distances used in this context are the *Hamming* distance, i.e., the number of differing characters between two sequences, or the *edit* distance[1], i.e., the number of single-character edits (insertions or deletions) required to change one word into the other.

In our case, we take the point of view that the model is veracious and it is the log trace that may have errors. As such, we define the alignment problem as finding the model run $\mathcal{V}$ that requires the least amount of single-character edits in the log trace Λ. We make use of the edit distance and alignment definitions of [6] which we rephrase here:

Definition 15. *The edit distance $d(u, v)$ between two words $u, v \in \Sigma^*$ is the minimal number of single-symbol additions and deletions needed to transform u to v.*

[1]Also known as *Levenshtein* distance without the substitution operation.

Definition 16. *An alignment of a log trace* $\Lambda = \langle \Lambda_1, \ldots, \Lambda_m \rangle \in L$ *to a run* $\mathcal{V} = \langle \mathcal{V}_1, \ldots, \mathcal{V}_n \rangle$ *of process model* $M = (P, T, F, m_0, m_f, \Sigma, A)$ *is a sequence of pairs* $\langle (\Lambda'_1, \mathcal{V}'_1), \ldots, (\Lambda'_p, \mathcal{V}'_p) \rangle$ *with* $p \leq m + n$ *such that each pair* $(\Lambda'_i, \mathcal{V}'_i)$ *is either:*

- (σ_i, t_i) *with* $\sigma_i = A(t_i)$ *for a synchronous move in the log trace and the run,*
- $(\sigma_i, \gg)$ *for a deletion in the log trace, or*
- $(\gg, t_i)$ *for an insertion in the log trace.*

An alignment between $\mathcal{V}$ *and* Λ *is optimal if it minimizes the number of occurrences of* $\gg$*. This minimal number of mismatches corresponds to the edit distance* $d(\Lambda, \mathcal{V})$*.*

Then, onto the encoding, to find the best match from run $\mathcal{V}$ to log trace Λ we define a two-dimensional matrix $\mathcal{T}$ with boolean variables $\tau_{j,i}$, with $j \in [|\Lambda|] = \{1, \ldots, |\Lambda|\}$ and $i \in [N]$. Thus, each row j corresponds to each symbol Λ_j, while each column i corresponds to each transition $\mathcal{V}_i$. The purpose of this matrix $\mathcal{T}$ is to represent the same path as in the distance matrix for the Levenshtein distance computation, although through a boolean encoding.

To encode the symbol Λ_j from Λ, we define variables $\lambda_{j,a}$ for $j \in [|\Lambda|]$, the index of symbols in the log trace, and $a \in \Sigma$, the symbol of an action that labels a transition t in the model M. Then, $\lambda_{j,a}$ is true iff $\Lambda_j = A(t) = a$.

Similarly, we need the transition $\mathcal{V}_i$ from $\mathcal{V}$, and for that we use the variables $\nu_{i,t}$ from the Petri net encoding. These variables already indicate that transition t is active at instant i. At the same time, these variables are what allow us to link the Petri net rules to the edit distance computation.

We frame the problem of finding $\mathcal{V}$ as finding the best route through $\mathcal{T}$ from cell $\tau_{1,1}$ (the first symbols to check from Λ and $\mathcal{V}$) to cell $\tau_{|\Lambda|,N}$ (the last symbols from Λ and $\mathcal{V}$ that should match). We only want to be true the $\tau_{j,i}$ variables that are part of the route, while all others should be false. Let's define the rules that the route should ensure:

$$\tau_{1,1} \wedge \tau_{|\Lambda|,N} \tag{8}$$

$$\bigwedge_{i \in [N]} \bigwedge_{j \in [|\Lambda|]} \tau_{j,i} \rightarrow (\tau_{j,i+1} \vee \tau_{j+1,i} \vee \tau_{j+1,i+1}) \tag{9}$$

$$\bigwedge_{i \in [N]} \bigwedge_{j \in [|\Lambda|]} \tau_{j,i} \rightarrow (\tau_{j,i-1} \vee \tau_{j-1,i} \vee \tau_{j-1,i-1}) \tag{10}$$

$$\bigwedge_{i \in [N]} \bigwedge_{j \in [|\Lambda|]} (\tau_{j,i} \wedge \tau_{j+1,i}) \rightarrow (\neg\tau_{j,i+1} \wedge \neg\tau_{j-1,i+1} \wedge \neg\tau_{j+1,i-1}) \tag{11}$$

$$\bigwedge_{i \in [N]} \bigwedge_{j \in [|\Lambda|]} (\tau_{j,i} \wedge \tau_{j,i+1}) \rightarrow (\neg\tau_{j+1,i} \wedge \neg\tau_{j-1,i+1} \wedge \neg\tau_{j+1,i-1}) \tag{12}$$

$$\bigwedge_{i \in [N]} \bigwedge_{j \in [|\Lambda|]} (\tau_{j,i} \wedge \tau_{j+1,i+1}) \rightarrow (\neg\tau_{j-1,i+1} \wedge \neg\tau_{j+1,i-1} \wedge \neg\tau_{j+2,i} \wedge \neg\tau_{j,i+2}) \tag{13}$$

As required, we fix the first and last elements in the route, $\tau_{1,1}$ and $\tau_{|\Lambda|,N}$, with Constraint 8. Then, Constraints 9 and 10 state that for a given $\tau_{j,i}$ to be active, then one of its successors and one of its predecessors should also be active. We restrict the possible successors of $\tau_{j,i}$ to be the next cell down ($\tau_{j+1,i}, \downarrow$), right ($\tau_{j,i+1}, \rightarrow$) or down-right ($\tau_{j+1,i+1}, \searrow$). This definition of successor implies the definition of predecessor (up, left or up-left). Finally, Constraints 11, 12 and 13 force the creation of a band of zeros surrounding the activated $\tau_{j,i}$ for the down, right and down-right successor cases, respectively. These constraints represent the general case for the central part of the matrix; special care has to be taken for the boundary cells, adjusting the constraints accordingly to avoid accessing out-of-bounds memory.

As an example, Figure 2 shows a couple of possible routes through the $\mathcal{T}$ matrix for the last log trace from Figure 1, attending to the previous constraints.

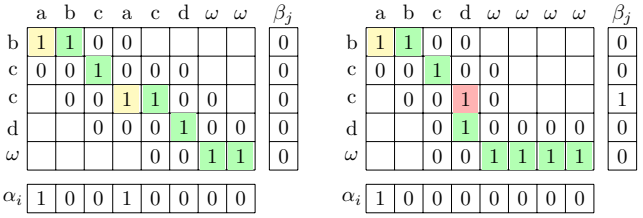

Fig. 2. The $\mathcal{T}$ matrix and full route (alignment) for the non-fitting log trace *bccd* and example labelled optimal runs *abcacd* (left) and *abcd* (right), both with edit distance 2, from Figure 1. Only the active $\tau_{j,i}$ cells and surrounding zeroed band are represented. All empty cells would be false, too. Green cells (diagonal movements) are match events between trace and run, yellow cells (horizontal movements) imply an insertion from run to trace and red cells (vertical movements) imply a deletion in the trace. Horizontal and vertical movements are also reflected in the α_i and β_j variables used in the optimization objective (see Constraints 21 and 22).

At this point, due to the zeroed band around the route through the matrix we avoid the possibility of additional paths, so all cells not pertaining to the route should be false, too. In other words, as the route through the matrix has to be right descending, no two different paths diverging from a cell and converging in any other below and to the right should be able to coexist. To enforce this effect by propagation and reduce search, we can further add AMO constraints to the up-right diagonals within the matrix ($\nearrow$):

$$\bigwedge_{k \in [|\Lambda|+N-1]} \left(\sum_{j \in [|\Lambda|], i \in [N] : j+i=k+1} \tau_{j,i} \right) \leq 1 \tag{14}$$

Constraint 14 will limit, for any up-right diagonal k, the possible cells being true to be only one at most. This should provide an early fail point when the solver decides τ's that would enable multiple paths, without requiring the completion of the principal route with the zeroed band.

Due to the "movement" rules in the $\mathcal{T}$ matrix, we can encounter three possible cases (as in Definition 16) for which we assign a meaning in terms of $\mathcal{V}$ and Λ matching (see Figure 2):

- If we have a vertical movement, e.g., $\tau_{j,i} \wedge \tau_{j+1,i}$, then symbols Λ_j and $A(\mathcal{V}_i)$ do not match and symbol Λ_j should be deleted from the log trace.
- If we have a horizontal movement, e.g., $\tau_{j,i} \wedge \tau_{j,i+1}$, then symbols Λ_j and $A(\mathcal{V}_i)$ do not match and symbol $A(\mathcal{V}_i)$ should be inserted into the log trace.
- If we have a diagonal movement, e.g. $\tau_{j,i} \wedge \tau_{j+1,i+1} \wedge \neg\tau_{j+1,i} \wedge \neg\tau_{j,i+1}$, then symbols Λ_j and $A(\mathcal{V}_i)$ match.

To help checking if transition $\mathcal{V}_i$ matches the symbol Λ_j, we define the test $[\mathcal{V}_i = \Lambda_j]$ as

$$\bigvee_{t \in T} \nu_{i,t} \wedge \lambda_{j,A(t)} \tag{15}$$

Now, to identify the three different cases and where they occur, we define variables α_i and β_j, with $i \in [N]$ and $j \in [|\Lambda|]$, respectively. We use these variables with the following constraints:

$$\bigwedge_{i \in [N]} \left(\bigvee_{j \in [|\Lambda|]} \tau_{j,i} \wedge [\mathcal{V}_i = \Lambda_j] \right) \leftrightarrow \neg\alpha_i \tag{16}$$

$$\bigwedge_{j < |\Lambda|} \left(\bigvee_{i \in [N]} \tau_{j,i} \wedge \tau_{j+1,i} \right) \leftrightarrow \beta_j \tag{17}$$

On the one hand, Constraint 16 ensures that α_i is false if we had a match between model run and log trace. Notice that, for a given i if α_i is true, then that means that $A(\mathcal{V}_i)$ could not be matched in the log trace and must be inserted. On the other hand, Constraint 17 activates β_j if symbol Λ_j could not be matched and should be removed from the log trace.

Related to Constraints 13 and 16, if we detect a match between Λ_j and $\mathcal{V}_i$ and $\tau_{j,i}$ is true, then we can directly enable the next position in the matrix ($\tau_{j+1,i+1}$) and force the alternatives ($\tau_{j+1,i}$ and $\tau_{j,i+1}$) to be 0 (again, explicit care has to be taken at the boundaries):

$$\bigwedge_{j \in [|\Lambda|]} \bigwedge_{i \in [N]} (\tau_{j,i} \wedge [\mathcal{V}_i = \Lambda_j]) \rightarrow (\tau_{j+1,i+1} \wedge \neg\tau_{j+1,i} \wedge \neg\tau_{j,i+1}) \tag{18}$$

In contrast, if a given $\tau_{j,i}$ is activated, but there is no match between trace and run (the negation of Formula 15, represented here as $[\mathcal{V}_i \neq \Lambda_j]$), we know that an insertion or a deletion has to be done, which corresponds to an horizontal or vertical movement in the matrix, respectively:

$$\bigwedge_{j \in [|\Lambda|]} \bigwedge_{i \in [N]} (\tau_{j,i} \wedge [\mathcal{V}_i \neq \Lambda_j]) \rightarrow (\tau_{j+1,i} \vee \tau_{j,i+1}) \tag{19}$$

Due to this constraint, only one of the successors denoted in the right hand side of the constraint (modified depending on boundary requirements) shall be active because of the rules of movement in the matrix and the zeroed band (in particular, Constraints 11 and 12). This will only propagate when one of these two successors is decided. Notice that this behaviour is also enforced by the diagonal AMO (Constraint 14) we set up to reduce search.

Now, to deal with runs and log traces of different sizes, and similar in spirit to [6], we augment the available transitions of the model M with a filler transition t labelled ω, i.e., $A(t) = \omega$, and do two things. First of all, we add a loop to this transition from the final marking m_f in the Petri net, so a run can be artificially as long as required just by firing this filler transition. Second of all, we append this filler transition as a last symbol in the log trace Λ (as if it was an End-Of-Line character). We will also require that once the filler transition is used in the run, all additional transitions should also be the filler:

$$\bigwedge_{i < N} \nu_{i,t=\omega} \rightarrow \nu_{i+1,t=\omega} \tag{20}$$

This strategy will make the run and the trace match at least one ω transition (the last character of Λ). Should the run require more transitions to reach a length N, all additional transitions will also be ω, which all will match with the log trace, due to the boundary modification required for Constraint 18. As an example, both runs in Figure 2 require a different number of filler transitions.

Our initial objective was to find the run $\mathcal{V}$ that most closely matches the log trace Λ, which in this encoding is translated to the one that has the lowest number of α_i and β_j variables active, i.e., the most symbols from Λ matched in $\mathcal{V}$. As a last step, we encode this objective function with the soft clauses:

$$\bigwedge_{i \in [N]} (\neg\alpha_i, 1) \tag{21}$$

$$\bigwedge_{j \in [|\Lambda|]} (\neg\beta_j, 1) \tag{22}$$

From the α_i and β_j variables of a resulting assignment, the edit distance will be $d(\Lambda, \mathcal{V}) = \sum_i \alpha_i + \sum_j \beta_j$, i.e., the sum of insertions (α_i) and deletions (β_j), and will precisely be the cost returned by the MaxSAT solver.

B. Multi-Alignments

Multi-alignments were generalized from the concept of alignments in [3], where the authors used them to cluster log traces around representative full runs of a model. In [5] multi-alignments are defined as

Definition 17. *Given a finite collection L of log traces and a model M, a multi-alignment of L to M is a full run $\mathcal{V} \in \mathcal{R}_M$ that minimizes $\sum_{\Lambda \in L} d(\Lambda, \mathcal{V})$.*

Our alignment encoding from Section IV-A can be directly extended to support this definition of multi-alignments. The only requirement we have is the necessity to duplicate the variables and clauses related to each of the log traces. In particular, for each $\Lambda \in L$ we will need variables $\lambda^{\Lambda}_{j,a}$, $\tau^{\Lambda}_{j,i}$, α^{Λ}_i and β^{Λ}_j, with $a \in \Sigma$, $j \in [|\Lambda|]$ and $i \in [|N|]$, plus the auxiliary variables for the regular encoding and typical transformations, as well as the corresponding clauses.

C. Anti-Alignments

Anti-alignments were presented in [4]; the authors introduced them to find highly deviating behaviour from a model, not found in the log traces. In [5] they are defined as

Definition 18. *Given a finite collection L of log traces and a model M, an anti-alignment is a full run $\mathcal{V} \in \mathcal{R}_M$ that maximizes $\sum_{\Lambda \in L} d(\Lambda, \mathcal{V})$.*

The encoding of anti-alignments is very similar to the multi-alignment version. As before, we have to duplicate variables and clauses related to the traces $\Lambda \in L$. In this case we also have to modify the objective function to find the model run $\mathcal{V}$ that is most different, i.e., has greatest distance, to the log traces $\Lambda \in L$; we have to change Constraints 21 and 22 for the next ones:

$$\bigwedge_{i \in [N]} (\alpha_i, 1) \tag{23}$$

$$\bigwedge_{j \in [|\Lambda|]} (\beta_j, 1) \tag{24}$$

These constraints work as anti-alignment for one log trace Λ. For a whole log L, we will also have to replicate them for all possible log traces $\Lambda \in L$, as we did in the multi-alignments.

V. Experimental Results

In this section we report on the experimental investigation conducted to assess the presented encodings.

We start by defining the used benchmarks, which include 3 artificial logs (M-models [21], Loan [22], BPM13 [23]) and 5 real-world logs (BPIC2012 [24], BPIC2013 [25], Traffic Fine Management [26], Sepsis [27], CCC19 [28]). Inductive Miner [29] was used for those instances that did not provide their own model. Table I provides information about model and log, where $|T|$ is the number of transitions, $|P|$ is the number of places and $|L|$ is the size of the log considered (number of traces). The run size column is the parameter for the size of run $\mathcal{V}$ and is computed from the size of the trace for alignments and from the largest trace considered in the case of multi and anti-alignments.

The environment of execution is a computer cluster with machines equipped with two Intel Xeon Silver 4110 at 2.1GHz and 96GB DDR4 main memory. All the experiments were executed with a timeout of 4h and a memory limit of 30GB. The complete MaxSAT solver used in all cases is RC2 [30], as such, if a benchmark provides a solution within the budgeted time, we can be sure it is optimal, e.g., in the case of alignments we obtain a model run with minimal α_i's and β_j's active or minimal edit distance to the log trace.

For our encoding into Partial MaxSAT, we use PM4Py [31] to handle logs and models in standard formats and OptiLOG [32] to create the formulas into common DIMACS format.

As our encoding is most similar to [5], we will use it as baseline in our comparison of resulting formula sizes and solving time for the defined benchmarks.

The theoretical complexity reported by [6] for full model runs of length n of a process model $M = (P, T, F, m_0, m_f, \Sigma, A)$ has size $O(n \times |F|)$, while the size of their edit distance formula is $O((n + m) \times n \times m)$, between a run of length n and a log trace of size m. In the case of multi- and anti-alignments, for a log L with maximum length of traces m, the total formula has a size of $O((n + m) \times n \times m \times |L| + n \times |F|)$. In our case, we have adapted their encoding for the Petri net, so the same size would be expected. Our edit distance is different as we do not explicitly encode it, our edit distance formula has a lower bound size of $O(n \times m + n + m)$, including α_i and β_j variables but excluding auxiliary variables from the regular encoding of AMO constraints. This size depends on the $\mathcal{T}$ matrix ($O(n \times m)$) plus the α_i ($O(n)$) and β_j ($O(m)$) vectors. As such, for multi- and anti-alignments, for a log L with maximum length of traces m, our total formula has an expected size of $O((n \times m + n + m) \times |L| + n \times |F|)$.

To check the effective difference in generated formula size, as well as solving time, we use the available implementation *da4py* from [5] and [6] and modify it to output the generated formulas (instead of directly solving them). We configure the instances in *da4py* with the optimization for multi-alignments and anti-alignments as described in the preceding section as sum of distances. Columns *da4py* and *ours* of Table I represent the resulting formula size of the DIMACS file in MB and the time used for the execution of the solver in seconds. Some caveats about this comparison: a) *da4py* depends on a maximum number of edits allowed for its encoding while *ours* does not; we compute this needed parameter from the number of edits that *our* approach reports, and b) to avoid great memory blowups on the formula generation for *da4py*, we have limited the number of edits to 20 (for example, our approach reports 50 edits for the multi-alignment of 10 traces of BPM13 A, or 130 edits for the anti-alignment of 50 traces of BPIC2012).

In the table, for instances that have a '-' in the time column the solver did not finish within the given timeout. For instances that have a '-' in the size column it means that the formula could not be generated within the allowed memory budget.

The difference in the size of the encodings as computed from the theoretical standpoint between *da4py* and *our* approach is corroborated. In the cases where *da4py* could compute the formula within the memory budget, that formula is on average 10 times bigger (1,75x for the multi-alignment in Loan Conf4 with 10 traces and 31.54x in the anti-alignment in BPIC2012 with 10 traces). On the performance side, the increase in the formula size for *da4py* indicates a much higher number of clauses and variables involved, which require more time for the solver to provide a solution. This becomes a trade-off argument: while our encoding supports more distance in smaller formula sizes and lower solving time, it can not put a limit on that edit distance, so *da4py* can become more flexible to the problem at hand (computing, for example, alignments with at most k edits) at the expense of more memory usage and computing time. Another aspect that *da4py* readily provides and has not been shown in the present comparison is the possibility of working with prefixes or non-complete traces; this functionality will be explored in our future work.

Additionally, we wanted to corroborate if the Diagonal AMO (Constraint 14) really provides the expected speed-up

TABLE I

RESULTS ON THE COMPUTATION OF CONFORMANCE ARTEFACTS: COMPARISON OF FORMULA SIZE AND SOLVING TIME. BEST RESULTS IN BOLD.

Instance	$\|T\|$	$\|P\|$	$\|L\|$	run size	max edits	da4py size (MB)	da4py time (s)	ours size (MB)	ours time (s)	ours-noAMO size (MB)	ours-noAMO time (s)
Alignments (averages over $\|L\|$ log traces)											
M1	39	40	50	17	8	7.43	4.74	1.55	**0.60**	**1.54**	0.75
M5	33	35	50	36	13	44.70	98.07	5.93	**96.66**	**5.86**	122.55
M8	15	17	50	9	5	5.48	0.43	0.84	16.68	**0.82**	**0.18**
Loan Conf1	9	10	50	7	5	0.26	0.18	**0.07**	**0.15**	**0.07**	**0.15**
Loan Conf4	8	7	50	7	5	0.21	0.18	**0.06**	**0.14**	**0.06**	**0.14**
BPM13 A	363	347	50	38	20	928.29	304.77	215.36	**109.62**	**215.30**	208.42
BPM13 B	317	317	50	36	20	1042.71	267.83	228.75	118.39	**228.14**	**97.62**
BPIC2012	47	22	50	10	8	41.65	1.41	6.36	60.78	**6.30**	**0.30**
BPIC2013 open problems	6	4	50	3	5	0.09	**0.11**	**0.02**	**0.11**	**0.02**	0.12
BPIC2013 closed problems	19	15	50	7	8	1.00	0.37	0.15	**0.13**	**0.14**	0.15
Traffic Fine Management	29	19	50	6	7	0.76	0.38	**0.15**	0.17	**0.15**	**0.16**
Sepsis	30	14	50	9	11	5.58	1.01	**0.64**	0.45	**0.64**	**0.25**
CCC19	31	27	20	55	20	138.91	28.61	20.05	48.37	**19.74**	**14.38**
Multi-alignments											
M1	39	40	10	20	13	165.55	1341.80	12.30	394.96	**12.16**	**295.96**
			50	26	5	725.32	-	82.53	-	**81.54**	-
M5	33	35	10	52	5	464.43	-	68.53	-	**67.52**	-
			50	52	5	-	-	389.64	-	**383.60**	-
M8	15	17	10	36	13	217.59	-	11.17	4678.40	**10.87**	**1487.75**
			50	62	5	-	-	110.39	-	**107.26**	-
Loan Conf1	9	10	10	7	5	2.41	0.77	0.61	**2.00**	**0.59**	2.29
			50	7	5	13.31	3.61	3.25	2.51	**3.16**	**0.88**
Loan Conf4	8	7	10	7	5	2.06	0.70	1.16	5.01	**0.54**	**0.25**
			50	7	5	11.46	2.96	3.02	1.42	**2.93**	**1.19**
BPM13 A	363	347	10	35	20	-	-	611.47	363.00	**610.79**	**249.69**
			20	40	20	-	-	1215.71	**1407.58**	**1214.08**	-
BPM13 B	317	317	10	51	20	-	-	942.76	**495.98**	**941.39**	-
			20	51	20	-	-	1576.83	**2116.51**	**1574.09**	10612.29
BPIC2012	47	22	10	60	5	984.06	-	66.00	-	**65.29**	-
			50	109	5	-	-	803.40	-	**794.17**	-
BPIC2013 open problems	6	4	10	8	6	2.82	0.96	0.42	**0.81**	**0.41**	0.94
			50	9	5	16.73	8.65	1.97	**6.50**	**1.91**	9.41
BPIC2013 closed problems	19	15	10	9	8	14.34	4.14	1.35	**2.08**	**1.33**	5.14
			50	14	5	118.08	**703.36**	10.65	-	**10.42**	8243.02
Traffic Fine Management	29	19	10	7	7	11.00	6.02	1.18	7.51	**1.16**	**1.83**
			50	7	7	57.73	14.71	5.21	**2.02**	**5.16**	5.06
Sepsis	30	14	10	23	14	181.55	5206.21	10.23	**1422.42**	**10.07**	1776.62
			50	25	5	551.10	-	54.08	-	**53.26**	-
CCC19	31	27	10	119	5	-	-	343.00	-	**337.25**	-
			20	119	5	-	-	682.79	-	**671.43**	-
Anti-alignments											
M1	39	40	10	20	20	167.04	39.04	12.30	**2.81**	**12.16**	2.83
			50	26	20	1488.79	317.64	82.53	**14.12**	**81.54**	17.34
M5	33	35	10	52	20	998.50	214.53	68.52	14.12	**67.52**	**13.99**
			50	52	20	-	-	389.63	**77.63**	**383.60**	78.07
M8	15	17	10	36	20	222.41	52.10	11.17	2.55	**10.88**	**2.47**
			50	62	20	-	-	110.38	23.14	**107.26**	**23.07**
Loan Conf1	9	10	10	7	12	5.22	1.67	0.61	0.40	**0.59**	**0.26**
			50	7	12	27.59	22.52	3.25	0.85	**3.16**	**0.81**
Loan Conf4	8	7	10	7	9	3.82	1.21	0.56	**0.28**	**0.54**	0.33
			50	7	9	21.51	6.37	3.02	0.86	**2.93**	**0.85**
BPM13 A	363	347	10	35	5	-	-	611.47	-	**610.79**	-
			20	40	20	-	-	1215.71	**213.76**	**1214.08**	221.23
BPM13 B	317	317	10	51	20	-	-	942.76	**1555.56**	**941.39**	1714.505
			20	51	20	-	-	1576.83	**1491.12**	**1574.09**	1646.68
BPIC2012	47	22	10	60	20	2082.12	452.88	66.00	15.04	**65.28**	**14.22**
			50	109	20	-	-	803.39	181.83	**794.17**	**152.23**
BPIC2013 open problems	6	4	10	8	13	5.18	1.57	**0.42**	0.22	**0.42**	**0.20**
			50	9	14	35.03	16.69	1.97	**0.55**	**1.91**	0.59
BPIC2013 closed problems	19	15	10	9	16	19.02	5.44	1.35	**0.41**	**1.33**	**0.41**
			50	14	20	242.46	1513.12	10.65	2.33	**10.42**	**2.29**
Traffic Fine Management	29	19	10	7	10	16.52	4.18	1.18	**0.39**	**1.16**	**0.39**
			50	7	9	75.81	18.57	5.21	1.30	**5.16**	**1.26**
Sepsis	30	14	10	23	20	183.53	41.96	10.23	2.38	**10.07**	**2.31**
			50	25	20	1145.32	250.92	54.07	11.34	**53.26**	**11.13**
CCC19	31	27	10	119	20	-	-	343.00	66.56	**337.24**	**65.66**
			20	119	20	-	-	682.79	124.17	**671.42**	**122.86**

due to the hypothesized early fail point or it is a constraint that could be removed. As such, we have also done the same experimentation disabling that constraint in the encoding; the results are provided in column *ours-noAMO* of Table I.

Discarding the AMO constraint obviously requires less clauses and auxiliary variables (2% less clauses and variables on average). However, it is surprising that only a third of the reported results in the table (23/65) have explicitly better timings using the Diagonal AMO than not using it (26/65). While the rationale behind the Diagonal AMO seems reasonable because it forces propagation, it is also true that introduces more variables that the solver can decide on (increasing the branching on the search) and it is, in fact, a redundant constraint: the movement rules on the $\mathcal{T}$ matrix and the zeroed band already enforce that no two routes from the top-left cell to the bottom-right one can coexist.

VI. Conclusions

We have presented a new Partial MaxSAT encoding for the edit distance computation of alignments, multi- and anti-alignments that reduces the memory requirements and outperforms in solving time for full traces in comparison with the state-of-the-art SAT encodings from [5] and [6].

One limitation that we will lift in future work is the requirement of a full log trace and run of the model, which imposes a memory problem for large logs. Additionally, the encoded objective as sum of distances (for multi- and anti-alignments) is a proxy for good solutions but, as reported by [6], can fail in some cases. In future work we intend to adapt a similar strategy to our present encoding as well as pursue further optimizations.

References

[1] J. Carmona, B. F. v. Dongen, A. Solti, and M. Weidlich, *Conformance Checking: Relating Processes and Models*. Springer International Publishing, 2018.

[2] A. Adriansyah, "Aligning observed and modeled behavior," Ph.D. dissertation, Mathematics and Computer Science, 2014.

[3] T. Chatain, J. Carmona, and B. van Dongen, "Alignment-based trace clustering," in *Conceptual Modeling*, H. C. Mayr, G. Guizzardi, H. Ma, and O. Pastor, Eds. Cham: Springer International Publishing, 2017, pp. 295–308.

[4] T. Chatain and J. Carmona, "Anti-alignments in conformance checking – the dark side of process models," in *Application and Theory of Petri Nets and Concurrency*, F. Kordon and D. Moldt, Eds. Springer International Publishing, 2016, pp. 240–258.

[5] M. Boltenhagen, T. Chatain, and J. Carmona, "Encoding conformance checking artefacts in SAT," in *Business Process Management Workshops*, C. Di Francescomarino, R. Dijkman, and U. Zdun, Eds. Cham: Springer International Publishing, 2019, pp. 160–171.

[6] ——, "Optimized SAT encoding of conformance checking artefacts," *Computing*, vol. 103, pp. 29–50, 2021.

[7] T. Chatain, M. Boltenhagen, and J. Carmona, "Anti-alignments—measuring the precision of process models and event logs," *Information Systems*, vol. 98, p. 101708, 2021.

[8] G. Winskel, "Petri nets, algebras, morphisms, and compositionality," *Information and Computation*, vol. 72, no. 3, pp. 197–238, 1987.

[9] M. de Leoni and A. Marrella, "Aligning real process executions and prescriptive process models through automated planning," *Expert Systems with Applications*, vol. 82, pp. 162–183, 2017.

[10] V. Bloemen, J. van de Pol, and W. M. van der Aalst, "Symbolically aligning observed and modelled behaviour," in *2018 18th International Conference on Application of Concurrency to System Design (ACSD)*, 2018, pp. 50–59.

[11] D. Reißner, R. Conforti, M. Dumas, M. La Rosa, and A. Armas-Cervantes, "Scalable conformance checking of business processes," in *OTM 2017 Conferences*, H. Panetto, C. Debruyne, W. Gaaloul, M. Papazoglou, A. Paschke, C. A. Ardagna, and R. Meersman, Eds. Cham: Springer International Publishing, 2017, pp. 607–627.

[12] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Scalable process discovery and conformance checking," *Software & Systems Modeling*, vol. 17, no. 2, pp. 599–631, May 2018.

[13] W. M. P. van der Aalst, "Decomposing petri nets for process mining: A generic approach," *Distributed and Parallel Databases*, vol. 31, no. 4, pp. 471–507, Dec 2013.

[14] J. Munoz-Gama, J. Carmona, and W. M. van der Aalst, "Single-entry single-exit decomposed conformance checking," *Information Systems*, vol. 46, pp. 102–122, 2014.

[15] F. Taymouri and J. Carmona, "A recursive paradigm for aligning observed behavior of large structured process models," in *Business Process Management*, M. La Rosa, P. Loos, and O. Pastor, Eds. Cham: Springer International Publishing, 2016, pp. 197–214.

[16] ——, "Computing alignments of well-formed process models using local search," *ACM Trans. Softw. Eng. Methodol.*, vol. 29, no. 3, Jun. 2020.

[17] M. Boltenhagen, T. Chatain, and J. Carmona, "Generalized alignment-based trace clustering of process behavior," in *Application and Theory of Petri Nets and Concurrency*, S. Donatelli and S. Haar, Eds. Cham: Springer International Publishing, 2019, pp. 237–257.

[18] C. Ansótegui and F. Manyà, "Mapping problems with finite-domain variables into problems with boolean variables," in *SAT 2004 - The Seventh International Conference on Theory and Applications of Satisfiability Testing, 10-13 May 2004, Vancouver, BC, Canada*, 2004, pp. 1–15.

[19] I. P. Gent and P. Nightingale, "A new encoding of alldifferent into SAT," in *International Workshop on Modelling and Reformulating Constraint Satisfaction*, 2004, pp. 95–110.

[20] G. S. Tseitin, *On the Complexity of Derivation in Propositional Calculus*. Springer Berlin Heidelberg, 1983, pp. 466–483.

[21] F. Taymouri and J. Carmona, "Model and event log reductions to boost the computation of alignments," in *Data-Driven Process Discovery and Analysis*, P. Ceravolo, C. Guetl, and S. Rinderle-Ma, Eds. Cham: Springer International Publishing, 2018, pp. 1–21.

[22] J. Buijs, "Loan application example," Apr 2013. [Online]. Available: https://data.4tu.nl/collections/Loan_application_example/5065460/1

[23] J. Munoz-Gama, "'conformance checking in the large' (bpm 2013)," Apr 2013. [Online]. Available: https://data.4tu.nl/articles/dataset/_Conformance_Checking_in_the_Large_BPM_2013_/12692969/1

[24] B. van Dongen, "Bpi challenge 2012," Apr 2012. [Online]. Available: https://data.4tu.nl/articles/dataset/BPI_Challenge_2012/12689204/1

[25] W. Steeman, "Bpi challenge 2013," Apr 2014. [Online]. Available: https://data.4tu.nl/collections/BPI_Challenge_2013/5065448/1

[26] M. M. de Leoni and F. Mannhardt, "Road traffic fine management process," Feb 2015. [Online]. Available: https://data.4tu.nl/articles/dataset/Road_Traffic_Fine_Management_Process/12683249/1

[27] F. Mannhardt, "Sepsis cases - event log," Dec 2016. [Online]. Available: https://data.4tu.nl/articles/dataset/Sepsis_Cases_-_Event_Log/12707639/1

[28] J. Munoz-Gama, R. R. de la Fuente, M. M. Sepúlveda, and R. R. Fuentes, "Conformance checking challenge 2019 (ccc19)," Feb 2019. [Online]. Available: https://data.4tu.nl/articles/dataset/Conformance_Checking_Challenge_2019_CCC19_/12714932/1

[29] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Discovering block-structured process models from event logs containing infrequent behaviour," in *Business Process Management Workshops*, N. Lohmann, M. Song, and P. Wohed, Eds. Cham: Springer International Publishing, 2014, pp. 66–78.

[30] A. Ignatiev, A. Morgado, and J. Marques-Silva, "RC2: an Efficient MaxSAT Solver," *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 11, no. 1, pp. 53–64, 2019.

[31] A. Berti, S. J. van Zelst, and W. M. P. van der Aalst, "Process mining for python (PM4Py): Bridging the gap between process-and data science," in *Proceedings of the ICPM Demo Track 2019, Aachen, Germany, June 24-26, 2019.*, 2019, p. 13–16.

[32] C. Ansótegui, J. Ojeda, A. Pacheco, J. Pon, J. M. Salvia, and E. Torres, "OptiLog: A Framework for SAT-based Systems," in *SAT 2021 - 24th International Conference on Theory and Applications of Satisfiability Testing, July 5-9th 2021, Barcelona, Spain*, 2021.