An A*-Algorithm for Computing Discounted Anti-Alignments in Process Mining

Mathilde Boltenhagen
Université Paris-Saclay,
CNRS, ENS Paris-Saclay, Inria,
Laboratoire Méthodes Formelles
Gif-sur-Yvette, France
boltenhagen@lsv.fr

Thomas Chatain
Université Paris-Saclay,
CNRS, ENS Paris-Saclay, Inria,
Laboratoire Méthodes Formelles
Gif-sur-Yvette, France
chatain@lsv.fr

Josep Carmona
Universitat Politècnica de Catalunya
Barcelona, Spain
jcarmona@cs.upc.edu

***Abstract*—Process mining techniques aim at analyzing and monitoring processes through event data. Formal models like Petri nets serve as an effective representation of the processes. A central question in the field is to assess the conformance of a process model with respect to the real process executions. The notion of *anti-alignment*, which represents a model run that is as distant as possible to the process executions, has been demonstrated to be crucial to measure precision of models. However, the only known algorithm for computing anti-alignments has a high complexity, which prevents it from being applied on real-life problem instances. In this paper we propose a novel algorithm for computing anti-alignments, based on the well-known graph-based A^* scheme. By introducing a discount factor in the edit distance used for the search of anti-alignments, we obtain the first efficient algorithm to approximate them. We show how this approximation is quite accurate in practice, by comparing it with the optimal results for small instances where the optimal algorithm can also compute anti-alignments. Finally, we compare the obtained precision metric with respect to the state-of-the-art metrics in the literature for real-life examples.**

***Index Terms*—Process Mining, Conformance Checking**

I. Introduction

As event data becomes a ubiquitous source of information, data science techniques represent an unprecedented opportunity to analyze and react to the processes that generate these data. *Process Mining* is an emerging field that contributes to this aim [1].

Process mining proposes algorithms for discovering process models (e.g., Petri nets) from event logs, or assessing the adherence of a process model in describing the traces found in an event log. The later is known as *Conformance Checking*, and is one of the central pillars of the field [2].

Comparing a possibly infinite language (i.e., the process model may accept an infinite set of runs) with respect to a finite language (the event log is a finite set of traces) makes conformance checking techniques challenging. The notion of anti-alignment [3], [4], however, aims at aiding this comparison, since it explores the process model to find a run that deviates the most with respect to all the traces in the event log. Anti-alignments can be used for several purposes, e.g., evaluating the precision/generalization of a process model with respect to an event log [5], or log-based simulation of a process model. For instance, for the model and log shown in Fig. 1, an optimal anti-alignment is $\langle b, e, d, \tau\rangle$ (where τ holds for silent activity).

Still, anti-alignments are hard to compute. The only published algorithm for computing alignments is grounded on encoding the problem as a SAT instance [6], thus making the algorithm not applicable for large, industrial problem instances.

This paper proposes a novel algorithm to fight the complexity of computing anti-alignments. We consider a rather simple, yet powerful idea that is motivated from the following use case: for certain processes, the costs associated to deviations at early stages of the process' execution are more important than the ones at the end. For instance, this situation appears in *knock-out processes* [7], where two outcomes are possible: OK or NOK. As the tasks of those processes aim at determining the final output, checks of the task are usually ordered in a growing order to allow fast decisions [8].

Having this in mind, one can instantiate an A^* algorithm to make the cost function exponentially biased to this use case: giving more importance (higher costs) to the deviations that occur in early stages, and exponentially reducing the costs as the search algorithm progresses. This *discounted* cost function has been introduced in [9] for alignment where the aim is to relate a process model to single log trace. The work showed a large reduction in search space for a small loss of quality. When applying the idea to anti-alignments, we tackle a different problem because the search focuses on the counterpart of alignments, i.e., deviations are observed, and it considers the whole log. However, the overall idea is similar. It provides a huge impact on the size of the search space required for the A^* search, since the cost asymmetry makes the search space to rapidly shrink after the first steps are made.

Although the resulting algorithm is not guaranteed to provide optimal anti-alignments in general (i.e., maximally deviating model runs), the obtained implementation represents the first feasible attempt to compute anti-alignments for real-life problem instances. Although approximating the problem, in practice it can produce optimal results: for instance, the new algorithm can also provide anti-alignment $\langle b, e, d, \tau\rangle$ for the example of Fig. 1.

The paper is organized as follows: next section contextualizes the problem. In Section III the necessary background is detailed. Sections IV and V present the discounted cost function and the A^* algorithm, respectively. In Section VI, we explain how we adapt the precision measure. Then in Section VII we report the experimental evaluation in different dimensions, whilst Section VIII concludes the paper.

II. Related Works

Anti-alignments have been introduced by [3] as the *dark side* of process models. The same year, the authors present in [5] how those conformance artefacts can be used for measuring *precision* but also *generalization*, i.e., two fundamental metrics still in elaboration in process mining [1]. Anti-alignments have been proposed for both the Hamming distance and the Levenshtein distance [4]. Nicely, anti-alignment based precision metrics satisfy the necessary axioms for a precision metric [10]. Levenshtein distance is preferred, since it provides a more precise characterization of a deviation. Anti-alignments are the counterpart of *alignments* [11]. Given a log trace, an alignment is a run of a process model as similar as possible in terms of distance. While many optimizations exist for alignment computation [12]–[14], none of them has been adapted to compute anti-alignments since for the later, the whole log and not only one single trace needs to be considered. Today, the only known implementation for Levenshtein based anti-alignments is a SAT encoding given by [6] which is limited to small instances.

Since the main use of anti-alignments is the computation of precision, we now provide an overview of the current methods for this in the field. The work of [15] is one of the first attempts to evaluate the precision of a process model with respect to an event log. It is grounded on comparing relation matrices from the model and log. Since it requires the full state-space exploration of the process model, it is only applicable to small models. In [16] deviations are estimated by the number of *escaping arcs*, i.e., runs of process models that deviate from the log. The state-space exploration of this method is bounded by the log behaviors. However, this work does not consider the size of the deviations, i.e., escaping arcs might cause large deviant behaviors.

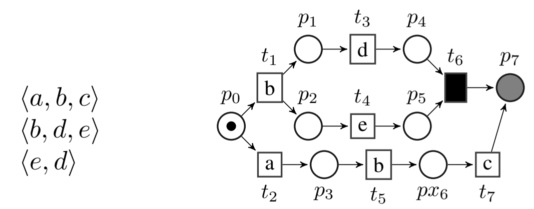
As reported in [10], the two works above fail at satisfying important axioms for a precision metric. We now turn the focus to recent proposals that, as the case for anti-alignment based precision, do satisfy the reference axioms for precision.

[17] transforms recorded and modeled behaviors into an abstraction called *directly follows* automaton, and compares their languages. By introducing the notion of *language quotients* according to a measure, a precision metric can be defined, which is the ratio of common sequences between the log and the model to the total model runs. To deal with infinite languages, they use the *topological entropy of languages*, detailed in [18]. However this technique is very strict when any sequence is shared by the log and the model. In [19], they add *skips* actions to make the approach more flexible. Finally, the work of [20] is based on *Markovian abstractions*. By comparing the k-th order Markovian abstraction of a process model against the respective one of an event log, a precision metric can be obtained. Both works above represent an interesting attempt to estimate precision, but due to being grounded in sometimes aggressive abstractions, they fail in generating concrete insights pinpointing the real deviations. In contrast, anti-alignments are not based on abstraction but instead in concrete model runs, that may serve as a concrete explanations for repairing precision problems in a process model.

III. Preliminaries

Event data can be represented by sequences of activities, i.e., traces, contained in logs defined as follows.

Definition 1 (Log)**.** *Let Σ be a set of activities. We define a* log *L as a finite multiset of log traces $\sigma \in \Sigma^*$.*



$\langle a, b, c \rangle$
$\langle b, d, e \rangle$
$\langle e, d \rangle$

(a) Log L (b) Petri Net N

Fig. 1: A log L and a Petri Net as Process Model

Thanks to discovery techniques [21], [22], one can obtain a process model from a log. Process models are often defined as Petri nets, which provide representations for causality, concurrency and loop behaviors.

Definition 2 (Petri Nets as Process Models)**.** *A* labeled Petri net *[23] 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 activity or as silent.*

Semantics.: The semantics of Petri nets are given in term of *firing sequences*. Given a node $x \in P \cup T$, we define its pre-set ${}^\bullet x \overset{\text{def}}{=} \{y \in P \cup T \mid (y, x) \in F\}$ and its post-set $x^\bullet \overset{\text{def}}{=} \{y \in P \cup T \mid (x, y) \in F\}$. A marking is an assignment of a non-negative integer to each place. A transition t is *enabled* in a marking m when all places in ${}^\bullet t$ are marked and can *fire* by removing a token from each place in ${}^\bullet t$ and putting a token to each place in $t^\bullet$. A marking m' is *reachable* from m if there is a sequence of firings $\langle t_1 \dots t_n \rangle$ that transforms m into m', denoted by $m[t_1 \dots t_n\rangle m'$. The set of reachable markings from m_0 is denoted by $[m_0\rangle$. A *full run* of a Petri net N is a firing sequence $m_0[t_1 \dots t_n\rangle m_f$ from the initial marking m_0 to the final marking m_f. A Petri net is *easy sound* [24] if it has at least one full run, i.e. m_f is reachable from m_0. In this paper we assume easy sound Petri nets and note $Runs(N)$ the set of full runs of a process model N.

Example 1. *Fig. 1 shows a log and a proposed, not optimal, process model. The initial marking m_0 is $\{p_0\}$ and the final marking is $\{p_7\}$. Full runs $Runs(N)$ of the process model are the firing sequences $\langle t_1, t_3, t_4, t_6\rangle$, $\langle t_1, t_4, t_3, t_6\rangle$ and $\langle t_2, t_5, t_7\rangle$ which give modeled behaviors $\langle b, e, d, \tau\rangle$, $\langle b, d, e, \tau\rangle$ and $\langle a, b, c\rangle$.*

To alleviate reading, we write the modeled sequences directly with the labels of the transitions in the rest of the paper.

Quality of process models is the core interest of conformance checking. In this paper, we focus on *precision*, a criterion that aims at quantifying how much extra modeled behavior is given by a process model and does not appear in the event log [1].

One method to formalize precision is based on the most deviant full runs of process models called *anti-alignments*.

Definition 3 (Anti-alignment)**.** *Given a log L and a model N, an anti-alignment is a full run of N given modeled sequence γ such that it maximizes the minimal distance $\min_{\sigma\in L} dist(\sigma, \gamma)$ to the log, where $dist$ is a distance between sequences.*

Common distance used in Process Mining is a version of the Levenshtein edit distance.

Definition 4 (Levenshtein Edit distance)**.** *The* Levenshtein Edit Distance *$\mathcal{L}(u, v)$ between two sequences u and $v \in \Sigma^*$ is the minimal number of edits (deletions or insertions) needed to transform u into v.*

$$\begin{cases} \mathcal{L}(\langle\rangle, \langle\rangle) = 0 \\ \mathcal{L}(u, \langle\rangle) = |u| \\ \mathcal{L}(\langle\rangle, v) = |v| \\ \mathcal{L}(a.u', b.v') = \mathcal{L}(u', v') & \textit{if } (a == b) \\ \mathcal{L}(a.u', b.v') = \min \begin{cases} \mathcal{L}(a.u', v) + 1, \\ \mathcal{L}(u, b.v') + 1 \end{cases} & \textit{otherwise.} \end{cases}$$

Example 2. *For log L of Fig. 1 and the presented process model N, the full run $\langle b, e, d, \tau\rangle$ is an anti-alignment and is at distance 1 or more to any log sequence.*

For models with executable loops, which produce long runs arbitrarily far from the associated log, no run maximizes the distance to the log. However, such long runs, although far from the log, are less relevant for measuring precision than shorter runs diverging from the log. Following this idea, a parameter ϵ is introduced in the definition of anti-alignments in order to penalize long runs as follow: $\sup\limits_{\gamma\in Runs(N)} \min\limits_{\sigma\in L} \frac{dist(\sigma,\gamma)}{(1+\epsilon)^{|\gamma|}}$. We denote with $\mathcal{L}_\epsilon(\gamma, \sigma) = \frac{\mathcal{L}(\gamma,\sigma)}{(1+\epsilon)^{|\gamma|}}$ the corresponding adaptation of Levenshtein distance.

Definition 5 (Anti-Alignment based Precision)**.** *For an event log L and a model N, ϵ-precision $P^\epsilon_{aa}(N, L)$ [4] is defined by:*

$$P^\epsilon_{aa}(N, L) \overset{\text{def}}{=} 1 - \sup_{\gamma\in Runs(N)} \min_{\sigma\in L} \Delta_\epsilon(\gamma, \sigma) \qquad (1)$$

with $\Delta_\epsilon(\gamma, \sigma) \overset{\text{def}}{=} \frac{\mathcal{L}_\epsilon(\gamma,\sigma)}{|\gamma|+|\sigma|}$ and $\epsilon \geq 0$.

Precision metric ranges between 0 (imprecise models) and 1 (precise models).

Example 3. *For $\epsilon = 0.01$ and L and N of Fig. 1, we have $P^\epsilon_{aa}(N, L) = 1 - \frac{\frac{1}{1.01^4}}{6} = 0.88$ for $\gamma = \langle b, e, d, \tau\rangle$ and $\sigma = \langle e, d\rangle$.*

IV. A Discounted Edit Distance for Prefix-based Alignment of Activities

Due to the complexity of recorded logs, process models tend to be large and contain a lot of choice, concurrency and loop behaviors. Naive exploration of the runs of a model has to consider a huge number of candidates for anti-alignments.

The aim of this paper is to reduce this exploration by using the discounted edit distance introduced in [9] which assigns higher costs to edits that appear at the beginning of the sequences.

Definition 6 (Discounted Edit Distance)**.** *We define the* Discounted Edit Distance *between two sequences u and v with discount parameter $\theta \geq 1$ by $\mathcal{D}_\theta(u, v) \overset{\text{def}}{=} \mathcal{D}^0_\theta(u, v)$ where:*

$$\begin{cases} \mathcal{D}^k_\theta(\langle\rangle, \langle\rangle) = 0 \\ \mathcal{D}^k_\theta(\langle\rangle, b.v') = \mathcal{D}^{k+1}_\theta(\langle\rangle, v') + \theta^{-k} \\ \mathcal{D}^k_\theta(a.u', \langle\rangle) = \mathcal{D}^{k+1}_\theta(u', \langle\rangle) + \theta^{-k} \\ \mathcal{D}^k_\theta(a.u', b.v') = \mathcal{D}^{k+2}_\theta(u', v') & \textit{if } (a == b) \\ \mathcal{D}^k_\theta(a.u', b.v') = \min \begin{cases} \mathcal{D}^{k+1}_\theta(u', v) + \theta^{-k} \\ \mathcal{D}^{k+1}_\theta(u, v') + \theta^{-k} \end{cases} & \textit{otherwise.} \end{cases}$$

Hence, insertions and deletions cost θ^{-k} where k refers to the position where they occur.

For $\theta = 1$, the Discounted Edit Distance is the Levenshtein distance. However, for larger values of θ, edits at the beginning of the sequences are more costly than edits at the end because of the exponent $-k$ based on the position of the edits.

By assigning higher costs to edits at the beginning of the sequences, the discounted parameter θ allows one to select efficiently prefixes of runs which may be continued to promising anti-alignments. Then revelant values for θ are slightly larger than 1 and close to 1 if the purpose is to approximate the Levenshtein edit distance.

Example 4. *For instance, for $\theta = 2$, an edit at position k costs more than the sum of all next edits of position $k' > k$. Let $L' = \{\langle b, c\rangle, \langle b, d, f\rangle\}$ be another log less conforming with process model N of Fig.1. The best anti-alignment for L' and N is $\langle a, b, c\rangle$ which is strongly more deviant in the beginning of the run with activity a. Its minimal distance to the log is $\theta^{-0} = 1$ and cannot be topped to another run despite summing edits. However, for lower values of the discounted parameter θ, like $\theta = 1.10$, the distance finds modeled behavior $\langle b, e, d, \tau\rangle$ as the furthest one from the log L', like when using the Levenshtein edit distance.*

V. An A*-based Algorithm for Computing Discounted Anti-Alignments

To get anti-alignments of a process model N and a log L using the discounted distance, we present an A*-based algorithm given in Alg. 1 that replays the firing sequences of the process model in order to find a full run γ such that :

$$\sup_{\gamma \in Runs(N)} \min_{\sigma \in L} \mathcal{D}_{\theta,\epsilon}(\gamma, \sigma) \tag{2}$$

where $\mathcal{D}_{\theta,\epsilon}(\gamma, \sigma) = \frac{\mathcal{D}_\theta(\gamma,\sigma)}{(1+\epsilon)^{|\gamma|}}$ to penalize long runs in case of loops in N.

The algorithm maintains a priority queue of prefixes of runs, implemented as a heap of tuples $\langle m, \gamma, d\rangle$, where γ is the prefix, m is the state that it reaches, and d is the priority with which it should be treated. This priority d is defined as the quantity:

$$h_{\theta,\epsilon}(\gamma, L) \overset{\text{def}}{=} \min_{\sigma \in L} \left(\mathcal{D}_{\theta,\epsilon}(\gamma, \sigma) + \frac{\theta^{-|\gamma|-|\sigma|}}{\theta - 1} \right) \tag{3}$$

that bounds from Eq. 2 the value $\min_{\sigma \in L} \mathcal{D}_{\theta,\epsilon}(\gamma', \sigma)$ that any full run γ' having γ as prefix can achieve.

New prefixes are obtained by firing the transitions of the process model (line 14 of Alg. 1) and the algorithm terminates when the queue is empty or no candidate prefix can improve the best value obtained so far (line 8).

Algorithm 1: Computation of Anti-Alignment using the Discounted Distance

```
Input  : $N = (P, T, F, \lambda, m_0, m_f)$: process model,
         $L$: log,
         $\theta$: discount parameter,
         $\epsilon$: long run limit parameter
1  $Q \leftarrow \{\langle m_0, \langle\rangle, h(\langle\rangle, L)\rangle\}$   // Heap of open states ordered
       by distance, maximum is placed on top
2  $\mathcal{B}_\gamma \leftarrow$ undefined                // Current best anti-alignment
3  $\mathcal{B}_\delta \leftarrow -\infty$                  // Current best distance to reach $m_f$
4  while $Q \neq \emptyset$                         // While not all states visited
5  do
6      $\langle m, \gamma, d\rangle \leftarrow Q.pop()$          // Next state maximizing $d$
7      if $d \leq \mathcal{B}_\delta$ then
8          break while     // No state is going to improve $\mathcal{B}_\delta$
9      if $m == m_f$ then
10         $\delta \leftarrow \min_{\sigma \in L} \mathcal{D}_{\theta,\epsilon}(\gamma, \sigma)$         // Exact distance $\delta$
11         if $\mathcal{B}_\delta < \delta$ then
12             $\mathcal{B}_\gamma \leftarrow \gamma$               // New best anti-alignment
13             $\mathcal{B}_\delta \leftarrow \delta$                      // Update distance
14     for $t \in T$ with $m[t\rangle m'$ do
15         $\gamma' \leftarrow \gamma \cdot t$                         // Get new prefix
16         $d' \leftarrow h(\gamma', L)$    // Get possible distance of $\gamma'$ to $L$
17         $Q \leftarrow Q.insert(\langle m', \gamma', d'\rangle)$          // Place new state
Output: $\mathcal{B}_\gamma$: best anti-alignment,
        $\mathcal{B}_\delta$: minimal distance of $\mathcal{B}_\gamma$ to $L$
```

A. Proof of optimality

Let us first prove the announced fact: for every full run γ' having γ as prefix, $h_{\theta,\epsilon}(\gamma, L) \geq \min_{\sigma \in L} \mathcal{D}_{\theta,\epsilon}(\gamma', \sigma)$. Let γ, $\gamma' = \gamma.u$ and σ, we show that $\mathcal{D}_{\theta,\epsilon}(\gamma, \sigma) + \frac{\theta^{-|\gamma|-|\sigma|}}{\theta-1} \geq \mathcal{D}_{\theta,\epsilon}(\gamma', \sigma)$. This is because, in order to edit γ' to σ, one can edit its prefix γ to σ (at cost $\mathcal{D}_{\theta,\epsilon}(\gamma, \sigma)$) and then delete the letters of u one by one. The first deletion costs $\theta^{-|\gamma|-|\sigma|-1}$, the second one $\theta^{-|\gamma|-|\sigma|-2}$... and the sum of these costs is bounded by $\sum_{i=1}^{\infty} \theta^{-|\gamma|-|\sigma|-i} = \frac{\theta^{-|\gamma|-|\sigma|}}{\theta-1}$.

The algorithm satisfies the following invariant, which holds before and after each iteration of the while loop: either the current best value is optimal or every optimal anti-alignment has a prefix in the queue Q. It is preserved at each iteration for the following reason. Let $\langle m, \gamma, d\rangle$ pop from Q and assume γ is the prefix of an optimal full run γ'. Either $\gamma' = \gamma$ (then $\min_{\sigma \in L} \mathcal{D}_{\theta,\epsilon}(\gamma, \sigma)$ is compared with $\mathcal{B}_\delta$) or γ' has a prefix of the form $\gamma.t$, which is queued. The while loop is broken (line 8) only when $d \leq \mathcal{B}_\delta$. As stated before, this implies that no γ' having γ as prefix will beat the current best value; and this also holds for all the other prefixes remaining in the queue since their value is smaller than d.

Termination: once $\mathcal{B}_\delta > 0$, any γ longer than $\frac{\log \frac{\theta}{(\theta-1)\mathcal{B}_\delta}}{\log(1+\epsilon)}$ popped from Q breaks the while loop. Indeed, this implies $(1+\epsilon)^{|\gamma|} \geq \frac{\theta}{(\theta-1)\mathcal{B}_\delta}$. Moreover, for every σ, $\mathcal{D}_\theta(\gamma, \sigma) \leq \sum_{i=0}^{\infty} \theta^{-i} = \frac{\theta}{\theta-1}$; hence $\mathcal{D}_{\theta,\epsilon}(\gamma, \sigma) \leq \mathcal{B}_\delta$ Since only finitely many prefixes are shorter, and no prefix is queued twice, the termination of the algorithm is guaranteed as soon as $\mathcal{B}_\delta > 0$, i.e. a full run $\gamma \notin L$ has been found.

Example 5. *This example illustrates the use of the fraction* $\frac{\theta^{-|\gamma|-|\sigma|}}{\theta-1}$ *in the definition of* $h_{\theta,\epsilon}$. *Let* $\theta = 1.10$ *and* $\epsilon = 0$ *for log* $L' = \{\langle b, c\rangle, \langle b, d, f\rangle\}$ *and model* N *of Fig.1. The open set* Q *contains, at some point, states* $\langle\{p_1, p_2\}, \langle b\rangle, d_b\rangle$ *and* $\langle\{p_3\}, \langle a\rangle, d_a\rangle$ *of concurrent prefixes* $\langle b\rangle$ *and* $\langle a\rangle$.

Let's try to consider only their distance to the log, i.e., $d_b = \min_{\sigma \in L} \mathcal{D}_{1.10,0}(\langle b\rangle, \sigma) = 0.83$ *and* $d_a = \min_{\sigma \in L} \mathcal{D}_{1.10,0}(\langle a\rangle, \sigma) = 2.73$. *Prefix* $\langle a\rangle$ *is the best current anti-alignment and the algorithm continues with this prefix (line 6) until the final marking. The final run is* $\mathcal{B}_\gamma = \langle a, b, c\rangle$ *and* $\mathcal{B}_\delta = 1$. *Prefix* $\langle b\rangle$ *is then forgotten at line 7 because its current distance is lower than* $\mathcal{B}_\delta$. *However, for* $\theta = 1.10$, *the optimal anti-alignment is indeed* $\langle b, e, d, \tau\rangle$. *By adding the fraction* $\frac{\theta^{-|\gamma|-|\sigma|}}{\theta-1}$ *in* d_b *and* d_a *given in function* h *at line 16 which prevents best suffixes,* $d_b = 7.51$ *and prefix* $\langle b\rangle$ *is now handled at line 7 and becomes the best anti-alignment.*

B. Complexity Analysis

The proof of termination above also gives us the time complexity of our algorithm: in the worst case, it explores all prefixes strictly shorter than $l \overset{\text{def}}{=} \frac{\log \frac{\theta}{(\theta-1)\mathcal{B}_\delta}}{\log(1+\epsilon)}$, and there may be up to $1+|T|+|T|^2+\cdots+|T|^l = \frac{|T|^{l+1}-1}{|T|-1}$ such prefixes. (In this expression $|T|$ can be replaced by the maximum branching degree of a reachable marking, i.e. the maximum number of transitions enabled together at a reachable marking.) Observe that l is larger when $\mathcal{B}_\delta$ is small, i.e. the algorithm takes more time when the model is precise.

C. A Heuristic for Alg. 1 to Further Reduce the Search Space

When models are dense in concurrency and loops, some markings are visited a large number of times. For instance, all possible combinations of a concurrent part finally reach the same marking. Because best prefixes are prioritized, a simple but efficient heuristic is to limit the number of times a marking can be reached. We propose a parameter μ for this purpose and observe very nice results in practice.

VI. Precision

The main purpose of finding anti-alignments is to compute precision of process models. The present paper provides an algorithm for finding anti-alignments not necessarily optimal, an artifact of reducing the search space. In return, the technique is able to work with real instances. Once an anti-alignment is found, one can use it to compute precision of its model by using the classical Levenshtein based precision as defined in Section III. The expression

$$\sup_{\gamma \in Runs(N)} \min_{\sigma \in L} \Delta_\epsilon(\gamma, \sigma) \tag{4}$$

of Def.5 is approximated by

$$\min_{\sigma \in L} \Delta_\epsilon(\gamma, \sigma) \tag{5}$$

for the run γ computed by Alg. 1. In Eq. (5), γ might not be an optimal anti-alignment as it appears in Eq. (4). Its minimal distance to the log can be smaller than the optimal one for a given log and model. Thus, using Eq. (5) in precision computation gives an upper-bound of the exact anti-alignment precision of Def.5. Nicely, we observe perfect matching results for some instances in practice.

VII. Experiments

The algorithm has been implemented in a copy of *pm4py* Python library[1]. In this section, we present a set of comparisons of the discounted distance based anti-alignment and precision with the state-of-the-art methods. The experiments have been run on a MacBook air 2017 model with a 1.8 GHz Intel ® Core$^{\text{TM}}$ i5 CPU and 8G RAM.

A. Comparison of the results obtained with the different heuristics

The aim of this first subsection is to present the risks and benefits of using the heuristics. For this purpose, we used the known artificial log $L'' = \{\langle A,B,D,E,I\rangle$, $\langle A,C,D,G,H,F,\rangle$, $\langle A,C,G,D,H,F\rangle$, $\langle A,C,H,D,F,I\rangle$, $\langle A,C,D,H,F,I\rangle\}$ and two of its associated models specifically chosen for showing the impact of the heuristics: the generating model and the flower model. The models can be found in [5] page 4, [20] and [4] page 10.

The generative model has a finite set of runs obtained by its several choice structures. By contrast, the flower model represents an infinite language of its transition labels which are all connected to the same place. We choose those models to present a normal case versus a complex and imprecise model.

Model	θ	Anti-alignment	P^ϵ_{aa}	Time (s)
Generating model	1.1	$\langle A,C,G,H,D,F,I\rangle$	0.928	0.02
	1.5	$\langle A,C,G,H,D,F,I\rangle$	0.928	0.01
	2.0	$\langle A,C,G,H,D,F,I\rangle$	0.928	0.01
Flower model	1.5	$\langle \tau,F,F,F,E,\tau\rangle$	0.400	29.85
	2.0	$\langle \tau,F,F,F,F,\tau\rangle$	0.372	4.67

(a) Complete Computation for Different Values of the Discounted Parameter θ for threshold limit $\epsilon = 0.01$

Model	μ	ϵ	Anti-alignment	P^ϵ_{aa}	Time (s)
Flower model	100	0.01	$\langle \tau,F,F,E,H,\tau\rangle$	0.400	2.50
	100	0.001	$\langle \tau,F,F,F,F,F,F,A,A,H,\tau\rangle$	0.302	4.61
	5	0.01	$\langle \tau,F,F,F,F,\tau\rangle$	0.372	0.07

(b) Reducing the Search Space with parameter μ for different ϵ values and $\theta = 1.5$

TABLE I: Heuristic Impacts on Anti-alignments by Using the Discounted Distance $\mathcal{D}_\theta$

In Tab. Ia, we show how changing the discounted parameter θ can improve the runtime. For large θ, prefixes cost more than suffixes, thus allowing the algorithm to considerably reduce the exploration. This aspect appears very clearly for the flower model.

The anti-alignments found for the flower model are not long because of the parameter ϵ set to 0.01. Then we observe that the found precision is quite high for this model, as it is highly imprecise. By setting ϵ to larger values, we would get longer anti-alignments, thus providing a more significant value for precision. However, due to the possible combinations of longer runs, the algorithm would blow up.

Tab. Ib aims at showing the benefit of the limit μ on the number of explorations of prefixes reaching the same marking. We see that for $\theta = 1.5$ and $\epsilon = 0.01$, the runtime of computing an anti-alignment for the flower model is divided by 10 when setting μ to 100. The runtime improvement by using μ is prominent and allows to explore longer runs (for instance Tab. Ib shows a result where $\epsilon = 0.001$). Finally, the last line of Tab. Ib presents an experiment where μ is very small and still provides a relevant anti-alignment (just a bit shorter).

There seems to be some parameter settings that work well for the type of instances we have used, but it still needs to be investigated how to adapt the parameters for arbitrary models and logs. We propose to set μ from 5 to 10 to limit the number of times the algorithm reaches the same state. This limitation is efficient for choice structures as the best paths are usually crossed first. For parallel structures, states are mainly different and this parameter does not help to tackle those complex patterns. Regarding θ, our experiments suggest to set it to a value between 1.1 (for better quality) and 1.5 (for faster result). Finally, ϵ can be set to 0.01 which already gives good approximations of model precision, but for more precise results we advise to drop this value to 0.001 which allows to observe longer runs.

[1] Available at https://github.com/BoltMaud/pm4py-core

B. Anti-alignments Comparison on Artificial Logs and Models

Now that we have presented the approximation impacts, we take advantage of them and compare our results to the optimal results for all artificial models associated to L'' where the SAT implementation of [4] can also compute the anti-alignments. To compute exact precision (column $E.P_{aa}^{\epsilon}$ of Tab. II) their algorithm solves several SAT formulas by increasing the size of the run until the anti-alignment converges. Because the process is slow, they propose another version in which one can set the size of the run if it is known. In Tab. II, we decided to compare runtimes of anti-alignment computation and we used the latter version. We set the size of the run to 11 which is the minimal size to get a full run for all the models.

Model	$E.P_{aa}^{\epsilon}$	Type	Anti-alignment	P_{aa}^{ϵ}	Time (s)
Generating model	0.928	$\mathcal{L}^n$	$\langle A, C, G, H, D, F, I \rangle$	0.923	16.51
		$\mathcal{D}_{\theta,\epsilon}$	$\langle A, C, G, H, D, F, I \rangle$	0.928	0.01
Single	1.000	$\mathcal{L}^n$	$\langle A, B, D, E, I \rangle$	1.000	9.92
		$\mathcal{D}_{\theta,\epsilon}$	$\langle A, B, D, E, I \rangle$	1.000	0.01
Flower model	0.295	$\mathcal{L}^n$	$\langle \tau, G, G, G, G, G, G, G, G, G\tau \rangle$	0.222	19.89
		$\mathcal{D}_{\theta,\epsilon}$	$\langle \tau, F, F, F, E, \tau \rangle$	0.400	0.03
Separate traces	1.000	$\mathcal{L}^n$	$\langle A, C, G, D, H, F, I \rangle$	1.000	42.26
		$\mathcal{D}_{\theta,\epsilon}$	$\langle A, B, D, E, I \rangle$	1.000	0.01
G,H in parallel	0.928	$\mathcal{L}^n$	$\langle A, C, D, G, \tau, F, I \rangle$	0.923	20.00
		$\mathcal{D}_{\theta,\epsilon}$	$\langle A, C, G, H, D, F, I \rangle$	0.928	0.04
G,H as self-loops	0.496	$\mathcal{L}^n$	$\langle A, C, G, D, G, G, G, G, G, F, I \rangle$	0.667	14.95
		$\mathcal{D}_{\theta,\epsilon}$	$\langle A, C, G^9, H, D, F, I \rangle$	0.631	0.15
D as self-loops	0.469	$\mathcal{L}^n$	$\langle A, B, D^7, E, I \rangle$	0.625	19.99
		$\mathcal{D}_{\theta,\epsilon}$	$\langle A, B, D^{10}, E, I \rangle$	0.588	0.17
All parallel	0.420	$\mathcal{L}^n$	$\langle \tau, I, E, F, D, H, C, B, A, G, \tau \rangle$	0.353	24.03
		$\mathcal{D}_{\theta,\epsilon}$	$\langle \tau, I, F, E, H, C, A, G, B, D, \tau \rangle$	0.525	4.08
C,F equal loop	0.845	$\mathcal{L}^n$	$\langle A, C, B, D, E, F, I \rangle$	0.833	18.01
		$\mathcal{D}_{\theta,\epsilon}$	$\langle A, C^7, B, D, E, F^7, I \rangle$	0.502	0.67
Round-robin	0.300	$\mathcal{L}^n$	$\langle \tau, E, F, G, H, I, A, B, C, D, \tau \rangle$	0.444	45.01
		$\mathcal{D}_{\theta,\epsilon}$	$\langle \tau, E, F, G, H, I, A, B, C, D, \tau \rangle$	0.502	0.01

TABLE II: Comparison of Anti-alignments and Precision on Artificial Log and Models. The discounted based method is noted by $\mathcal{D}_{\theta,\epsilon}$ with $\theta = 1.5$, $\epsilon = 0.01$ and $\mu = 10$ and $\mathcal{L}^n$ is the fastest Levenshtein version of [4] where the maximal size of the run $n = 11$. Column $E.P_{aa}^{\epsilon}$ shows the optimal anti-alignment precision where $\epsilon = 0.01$

Column P_{aa}^{ϵ} gives the approximations of the precision given by both the shorter version of [4] (lines $\mathcal{L}^n$ for Levenshtein with a size of the run n to 11) and the proposed algorithm of this paper (lines $\mathcal{D}_{\theta,\epsilon}$ where $\theta = 1.5$ and $\epsilon = 0.01$ and heuristic $\mu = 10$).

Observations are significant: the novel algorithm runs much faster and obtains, in most times, the optimal results. Differences between the found anti-alignments can be explained by the parameters. For instance, by using the flower model, our discounted version $\mathcal{D}_{\theta,\epsilon}$ returns a shorter run due to parameter μ which strictly reduces the number of times the algorithm can reach a marking.

The big difference observed for the model entitled *C,F equal loop*, for which our algorithm finds a much worse anti-alignment, is due to the fact that this model is not *safe*[2], i.e., transitions labeled with C and F output a token in their input place, thus allowing running the transitions again. However, the SAT implementation does not consider unsafe Petri nets (it restricts their behavior to runs visiting only safe markings), and, hence, cannot guarantee optimality for this model. The precision obtained with $\mathcal{D}_{\theta,\epsilon}$ is then more accurate than $E.P_{aa}^{\epsilon}$.

Finally, we observe that the analysis of the All parallel model, whose transitions are all concurrent, is much slower than the processing of all other models by using our A* scheme. In this model, the limit μ does not have any impact because the markings are different for all combinations of firing transitions. Also $\theta = 1.5$ is not too strict and many prefixes are explored. However, the runtime for this model is still faster than the fastest version of [4].

C. Precision of Real-life Models

In this section, we present, for the first time the application of our anti-alignment based precision on real-life models, whose description can be found in Tab. III. Those models have been discovered from the prototypes of the logs defined by [25] and by using the inductive miner (IM) [22] and the split miner (SM) [21][3]. We recall that the previous SAT implementation of anti-alignment could not deal with entire large logs due to the complexity of the encoding.

Log	$\|L\|$	$\|\Sigma\|$	Model Miner	$\|T\|$	$\|P\|$	$\|F\|$
BPI2012	13087	24	IM	34	24	68
			SM	30	23	60
BPI2019	251734	42	IM	18	13	38
			SM	13	10	26
BPI2020$_{dd}$	10500	17	IM	15	11	32
			SM	14	9	28
BPI2020$_{rp}$	6886	19	IM	31	26	74
			SM	23	12	46

TABLE III: Real-life Logs and Models Description.

We show both computation time and precision measure in Tab. IV and compare our work to the ETC measure defined by [11], the MAP^3 measure of [20] and the EMP measure proposed by [19], all introduced in the related works.

Log	Model	Precision				Runtime (s)			
		ETC	MAP^3	EMP	P_{aa}^{ϵ}	ETC	MAP^3	EMP	P_{aa}^{ϵ}
BPI2012	IM	0.561	0.492	0.602	0.761	513.28	6.94	44.84	79.73
	SM	0.915	0.196	0.538	0.753	438.65	6.98	27.50	176.69
BPI2019	IM	–	1.000	0.468	0.934	–	42.26	421.42	727.86
	SM	–	0.780	0.903	0.950	–	33.11	331.84	219.74
BPI2020$_{dd}$	IM	0.636	0.472	0.804	0.868	0.83	3.63	7.03	0.38
	SM	0.953	0.040	0.861	0.894	0.76	4.17	3.90	1.04
BPI2020$_{rp}$	IM	0.346	0.074	0.319	–	1.64	3.58	13.88	–
	SM	0.815	0.017	0.780	0.604	0.64	4.39	9.22	0.59

TABLE IV: Real-life Logs and Models Precision where the Anti-alignment Precision P_{aa}^{ϵ} is found with Discounting Anti-alignments and $\theta = 2$, $\epsilon = 0.01$, $\mu = 5$

We remark than any precision measure agrees for all the inputs which strengthens the interest of research about this metric in process mining. The MAP^3 measure finds a perfectly precise model, i.e., the IM model of BPI2019 log. Nicely, our method has a similar precision, 0.934, but found an anti-alignment for this model which can figures the language difference between the log and model. The EMP measure is stricter with a precision of 0.468. For this log, ETC precision

[2] A Petri net is safe if every place can have at most one token in any reachable marking.

[3] The used models are available at https://github.com/BoltMaud/A-Discounted-Cost-Function-for-Fast-Alignments-of-Business-Processes-Sources/tree/main/model

of [24] lasts more than several hours so we stopped the process. This method uses alignments whose complexity is related to the log complexity. As we can see in Tab. III, the BPI2019 log is the more complex log with 251 734 cases and 42 different type of activities.

Our method also has to deal with log complexity. More than 75% of our runtime presented in Tab. IV is used to compute the discounted distances. For the IM model of BPI2020$_{rp}$, our method also takes several hours. This model contains two parallel structures of 10 and 2 transitions, which make the search space explode. However, our method even gives the fastest runtime in several cases.

Besides runtimes, it is not obvious to conclude which precision metric is more valuable. However, as a conclusion of this section, we want to point out that, in order to understand the precision of process models, a human requires more than a numerical value. We believe that anti-alignment is the added value of our approach because it provides explanation for the imprecision of models. In the same way as in model checking a counterexample trace is used to show the violation of a property for a particular system, anti-alignments show traces that the user can understand and which show a model behavior that justifies a particular precision problem. In the extreme case, a very imprecise model will be easily noticed through its largely deviating anti-alignments.

VIII. Conclusion

In this paper, we show the advantage of a discounted distance to reduce the search space for finding anti-alignment of a process model w.r.t. an event log. Thanks to an A*-based algorithm that prioritizes promising prefixes, we find the optimal anti-alignments by using discounted costs slightly higher than 1 which, in practice, approximates well the Leventshein-based anti-alignments. We show through several experiments that our algorithm outperforms state-of-the-art methods for computing anti-alignments. Moreover, we see, for the first time, anti-alignment based precision for real-life instances.

In the real-life experiments, our algorithm hits a limit for a process model exhibiting high concurrency. Future work should target heuristics for this specific situation.

References

[1] W. Van Der Aalst, “Data science in action,” in *Process mining*. Springer, 2016.

[2] J. Carmona, B. F. van Dongen, A. Solti, and M. Weidlich, *Conformance Checking - Relating Processes and Models*. Springer, 2018. [Online]. Available: https://doi.org/10.1007/978-3-319-99414-7

[3] 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 - 37th International Conference, PETRI NETS 2016, Toruń, Poland, June 19-24, 2016. Proceedings*, 2016, pp. 240–258.

[4] T. Chatain, M. Boltenhagen, and J. Carmona, “Anti-alignments—measuring the precision of process models and event logs,” *Information Systems*, p. 101708, 2020.

[5] B. F. van Dongen, J. Carmona, and T. Chatain, “A unified approach for measuring precision and generalization based on anti-alignments,” in *International Conference on Business Process Management*. Springer, 2016, pp. 39–56.

[6] M. Boltenhagen, T. Chatain, and J. Carmona, “Optimized sat encoding of conformance checking artefacts,” *Computing*, pp. 1–22, 2020.

[7] W. M. van der Aalst, “Re-engineering knock-out processes,” *Decision Support Systems*, vol. 30, no. 4, pp. 451–468, 2001.

[8] H. A. Reijers and S. L. Mansar, “Best practices in business process redesign: an overview and qualitative evaluation of successful redesign heuristics,” *Omega*, vol. 33, no. 4, pp. 283–306, 2005.

[9] M. Boltenhagen, T. Chatain, and J. Carmona, “A discounted cost function for fast alignments ofbusiness processes,” in *International Conference on Business Process Management*. In Press, 2021.

[10] N. Tax, X. Lu, N. Sidorova, D. Fahland, and W. M. P. van der Aalst, “The imprecisions of precision measures in process mining,” *Inf. Process. Lett.*, vol. 135, pp. 1–8, 2018. [Online]. Available: https://doi.org/10.1016/j.ipl.2018.01.013

[11] A. Adriansyah, J. Munoz-Gama, J. Carmona, B. F. van Dongen, and W. M. van der Aalst, “Alignment based precision checking,” in *International Conference on Business Process Management*. Springer, 2012, pp. 137–149.

[12] W. L. J. Lee, H. Verbeek, J. Munoz-Gama, W. M. van der Aalst, and M. Sepúlveda, “Recomposing conformance: Closing the circle on decomposed alignment-based conformance checking in process mining,” *Information Sciences*, vol. 466, pp. 55–91, 2018.

[13] D. Reißner, R. Conforti, M. Dumas, M. La Rosa, and A. Armas-Cervantes, “Scalable conformance checking of business processes,” in *OTM Confederated International Conferences” On the Move to Meaningful Internet Systems”*. Springer, 2017, pp. 607–627.

[14] F. Taymouri and J. Carmona, “Computing alignments of well-formed process models using local search,” *ACM Trans. Softw. Eng. Methodol.*, vol. 29, no. 3, pp. 15:1–15:41, 2020. [Online]. Available: https://doi.org/10.1145/3394056

[15] A. Rozinat and W. M. Van der Aalst, “Conformance checking of processes based on monitoring real behavior,” *Information Systems*, vol. 33, no. 1, pp. 64–95, 2008.

[16] J. Munoz-Gama *et al.*, *Conformance checking and diagnosis in process mining*. Springer, 2016.

[17] A. Polyvyanyy, A. Solti, M. Weidlich, C. D. Ciccio, and J. Mendling, “Monotone precision and recall measures for comparing executions and specifications of dynamic systems,” *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 29, no. 3, pp. 1–41, 2020.

[18] T. Ceccherini-Silberstein, M. Antonio, and F. Scarabotti, “On the entropy of regular languages,” *Theoretical computer science*, vol. 307, no. 1, pp. 93–102, 2003.

[19] A. Kalenkova and A. Polyvyanyy, “A spectrum of entropy-based precision and recall measurements between partially matching designed and observed processes,” in *International Conference on Service-Oriented Computing*. Springer, 2020, pp. 337–354.

[20] A. Augusto, A. Armas-Cervantes, R. Conforti, M. Dumas, M. La Rosa, and D. Reißner, “Abstract-and-compare: A family of scalable precision measures for automated process discovery,” in *International Conference on Business Process Management*. Springer, 2018, pp. 158–175.

[21] A. Augusto, R. Conforti, M. Dumas, M. La Rosa, and A. Polyvyanyy, “Split miner: automated discovery of accurate and simple business process models from event logs,” *Knowledge and Information Systems*, vol. 59, no. 2, pp. 251–284, 2019.

[22] S. J. Leemans, D. Fahland, and W. M. van der Aalst, “Discovering block-structured process models from event logs containing infrequent behaviour,” in *International conference on business process management*. Springer, 2013, pp. 66–78.

[23] T. Murata, “Petri nets: Properties, analysis and applications,” *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–574, Apr. 1989.

[24] A. Adriansyah, B. F. van Dongen, and W. M. van der Aalst, “Memory-efficient alignment of observed and modeled behavior,” *BPM Center Report*, vol. 3, pp. 1–44, 2013.

[25] M. Fani Sani, M. Boltenhagen, and W. van der Aalst, “Prototype selection based on clustering and conformance metrics for model discovery,” in *International Conference on Business Process Management Workshops*. Springer, 2020.