Generating Counterfactual Explanations Under Temporal Constraints

Andrei Buliga[1,2*], **Chiara Di Francescomarino**[3*], **Chiara Ghidini**[2*], **Marco Montali**[2*], **Massimiliano Ronzani**[1*]

[1]Fondazione Bruno Kessler, Via Sommarive, 18, POVO - 38123, Trento, Italy
[2]Free University of Bozen-Bolzano, via Bruno Buozzi, 1 - 39100, Bozen-Bolzano, Italy
[3]University of Trento, Via Sommarive, 9, 38123 Trento, Italy
{abuliga, mronzani}@fbk.eu, c.difrancescomarino@unitn.it, {chiara.ghidini, marco.montali}@unibz.it

Abstract

Counterfactual explanations are one of the prominent eXplainable Artificial Intelligence (XAI) techniques, and suggest changes to input data that could alter predictions, leading to more favourable outcomes. Existing counterfactual methods do not readily apply to temporal domains, such as that of process mining, where data take the form of traces of activities that must obey to temporal background knowledge expressing which dynamics are possible and which not. Specifically, counterfactuals generated off-the-shelf may violate the background knowledge, leading to inconsistent explanations. This work tackles this challenge by introducing a novel approach for generating *temporally constrained counterfactuals*, guaranteed to comply by design with background knowledge expressed in Linear Temporal Logic on process traces (LTL_p). We do so by infusing automata-theoretic techniques for LTL_p inside a genetic algorithm for counterfactual generation. The empirical evaluation shows that the generated counterfactuals are temporally meaningful and more interpretable for applications involving temporal dependencies.

Code — https://github.com/abuliga/AAAI2025-temporal-constrained-counterfactuals

1 Introduction

State-of-the-art Machine Learning efforts prioritise accuracy using ensemble and deep learning techniques, but their complexity makes their input-output mappings difficult to interpret. To address this, eXplainable Artificial Intelligence (XAI) techniques have emerged, which aid in the interpretation of predictions and promoting the adoption of advanced models (Verma, Dickerson, and Hines 2020; Guidotti 2022; Dandl et al. 2020; Beckh et al. 2023).

Counterfactual explanations (Verma, Dickerson, and Hines 2020; Guidotti 2022) are a key eXplainable Artificial Intelligence (XAI) technique. They provide insights into which changes should be applied to an input instance to alter the outcome of a prediction. Such explanations are hence particularly valuable for users who need to understand how different attributes or actions might influence an outcome

of interest. State-of-the-art counterfactual generation methods often rely on optimisation techniques to find minimal changes to inputs leading to altering the predicted outcome.

Existing methods do not readily apply to temporal domains, such as that of process mining (van der Aalst 2016), where data of interest consists of traces of activities generated by executing a business/work process or a plan. Such so-called *process traces* (Fionda and Greco 2018) are increasingly used in key domains like healthcare, business, and industrial processes, where the sequencing of activities is central. The main issue in these settings is that not all sequencings of activities make sense: traces are typically subject to temporal constraints, that is, must comply with temporal background knowledge expressing which dynamics are possible and which not. E.g., in a healthcare process a patient may enter triage only upon giving privacy consent.

Systematic solutions for generating counterfactual explanations that comply with temporal background knowledge are still lacking (Buliga et al. 2023). This is a problem in areas such as Predictive Process Monitoring (PPM) (Di Francescomarino and Ghidini 2022), a widely established framework in process mining. Here, black-box predictive models are typically used to predict the outcome of executions of business and work processes, often making interpretability a challenge. Integrating counterfactual explanations into such monitoring frameworks can enhance understanding by providing alternative trace executions to reach a more favourable outcome, only upon guaranteeing that such explanations make sense, that is, comply with temporal background knowledge.

To tackle this open problem, we propose a novel optimisation based approach for generating *temporally constrained counterfactual traces*, that is, counterfactual process traces that are guaranteed to comply with temporal background knowledge. To express such knowledge, we take the natural choice of adopting LTL_p, a temporal logic specifically designed for process traces (Fionda and Greco 2018), starting from the well established formalism of linear temporal logic on finite traces (De Giacomo and Vardi 2013). The backbone of our approach consists in infusing automata-theoretic techniques for $\text{LTL}_p/\text{LTL}_f$ (De Giacomo et al. 2022) within a Genetic Algorithm (GA) body. In particular, given an LTL_p formula φ and a process trace τ satisfying φ, we introduce different strategies that inspect the automaton of φ to suit-

ably constrain the mutations introduced by the GA to alter τ towards counterfactual generation, so as to guarantee that the mutated version continues to satisfy φ.

An empirical evaluation on real-world and synthetic datasets demonstrates the effectiveness of our approach. Our results indicate that incorporating temporal background knowledge significantly improves the quality of counterfactual explanations, ensuring they satisfy LTL_p formulae of interest while satisfying general counterfactual desiderata.

2 Running Example

Consider a scenario in which an estate agency is using a PPM system to forecast the renting of flats to its customers during the application and negotiation process. The system takes in input an ongoing trace and forecasts **Accept** in case of a successful application and negotiation and **Fail** otherwise. For example, consider the ongoing trace τ_1, for which the predictive system forecasts a negotiation failure:

$\tau_1 = $ APPLY, AUT-CHK, MAN-CHK, PHONE, OK, OFFER, PHONE, BOOK

The trace starts with the customer submitting the application (APPLY) and the agency running an automatic check (AUT-CHK). Following a failure, a manual check (MAN-CHK) is done, and the customer is informed by phone (PHONE) of the application status, which is — in this case — accepted as valid (OK). An offer is created (OFFER) and communicated to the customer by phone (PHONE). The customer then asks for an appointment (BOOK) to discuss the offer.

To understand the reasons behind the prediction of **Fail** for τ_1, the estate agency intends to obtain counterfactual explanations for τ_1, that is, ongoing traces that suggest how to modify τ_1 so that a successful negotiation is predicted. Examples are reported below, where SEND-DOC indicates that the customer sends the required documents to the estate agency, and SMS (resp., EMAIL) captures that the agency informs the customer via sms (resp., e-mail):

$\tau_{c_1} = $ APPLY, <u>MAN-CHK</u>, <u>AUT-CHK</u>, PHONE, OK, OFFER, PHONE, SEND-DOC
$\tau_{c_2} = $ APPLY, AUT-CHK, MAN-CHK, PHONE, OK, OFFER, PHONE, SEND-DOC
$\tau_{c_3} = $ APPLY, AUT-CHK, <u>PHONE</u>, OK, OFFER, <u>SMS</u>, SEND-DOC
$\tau_{c_4} = $ APPLY, AUT-CHK, PHONE, PHONE, OK, OFFER, PHONE, SEND-DOC, BOOK
$\tau_{c_5} = $ APPLY, AUT-CHK, MAN-CHK, <u>PHONE</u>, OK, OFFER, <u>EMAIL</u>, SEND-DOC

We assume that counterfactual explanations $\tau_{c_1}, \ldots, \tau_{c_5}$ are properly built using available activities in the application and negotiation process. Nonetheless, some of them must be ruled out when considering how the agency operates. In our example, the agency has operational rules that state that: (i) within a negotiation, an automated check must eventually be conducted, and in case also a manual check is done, it can only be done after the automated one; and (ii) the agency always informs the applicant with the same communication mode (i.e., it is not possible to have, in the same negotiation, communications done with distinct modes). These two rules can be expressed using LTL_p (recalled next) as follows:

$\varphi_{chk} = (\neg\text{MAN-CHK}) \ \mathsf{U} \ \text{AUT-CHK}$
$\varphi_{comm} = \neg((\mathsf{F} \ \text{PHONE}) \wedge (\mathsf{F} \ \text{SMS})) \wedge \neg((\mathsf{F} \ \text{PHONE}) \wedge (\mathsf{F} \ \text{EMAIL})) \wedge$
$\qquad\qquad \neg((\mathsf{F} \ \text{SMS}) \wedge (\mathsf{F} \ \text{EMAIL}))$

It turns out that, among $\tau_{c_1}, \ldots, \tau_{c_5}$, τ_{c_1} violates φ_{chk},

while τ_{c_3} and τ_{c_5} violate φ_{comm}. The activities causing the violations are underlined in τ_{c_1}, τ_{c_3} and τ_{c_5}.

All in all, dealing with the presented scenario challenges state-of-the-art counterfactual generation approaches. In fact, when these approaches are applied in an out-of-the-box way, they cannot incorporate the background knowledge captured in φ_{chk} and φ_{comm}. What we need is a technique that only considers, as counterfactual traces, those that ensure to respect the temporal background.

3 Background

We overview here the essential background for the paper.

3.1 Linear Temporal Logic on Process Traces

Linear temporal logics, that is, temporal logics predicating on traces, are the most natural choice to express background knowledge in our setting. Traditionally, traces are assumed to have an infinite length, as witnessed by the main representative of this family of logics, namely LTL (Pnueli 1977). In several application domains, such as those mentioned in §1 and §2, the dynamics of the system are more naturally captured using unbounded, but *finite*, traces (De Giacomo, De Masellis, and Montali 2014). This led to LTL *over finite traces* (LTL_f) (De Giacomo and Vardi 2013), which adopts the syntax of LTL but interprets formulae over finite traces. In our work, we are specifically interested in a variant of LTL_f where propositions denote atomic activities constituting the basic building blocks of a process, and where each state indicates which atomic activity has been executed therein. This logic has been termed LTL *on process traces* (LTL_p) in (Fionda and Greco 2018), which we follow next.

Fix a finite set Σ of activities. A *(process) trace* τ over Σ is a finite, non-empty sequence $a_1, \ldots, a_n$ over Σ, indicating which activity from Σ is executed in every instant $i \in \{1, \ldots, n\}$ of the trace. The length n of τ is denoted $len(\tau)$. For $i < len(\tau)$, $\tau(i)$ denotes the activity a_i executed at instant i of τ, and $\tau(:i)$ the prefix $a_1, \ldots, a_i$ of τ.

An LTL_p formula φ is defined according to the grammar

$$\varphi ::= a \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \mathsf{X}\varphi \mid \varphi_1 \ \mathsf{U} \ \varphi_2, \text{ where } a \in \Sigma.$$

Intuitively, X is the *(strong) next* operator: $\mathsf{X}\varphi$ indicates that the next instant must be within the trace, and φ is true therein. U is the *until* operator: $\varphi_1 \ \mathsf{U} \ \varphi_2$ indicates that φ_2 is true now or in a later instant j of the trace, and in every instant between the current one and j excluded, φ_1 is true. Formally, given an LTL_p formula φ, a process trace τ, and an instance $i \in \{1, \ldots, len(\tau)\}$, we inductively define that φ is true in instant i of τ, written $\tau, i \models \varphi$, as:

$$\begin{aligned}
\tau, i &\models a && \text{if } \tau(i) = a \\
\tau, i &\models \neg\varphi && \text{if } \tau, i \not\models \varphi \\
\tau, i &\models \varphi_1 \vee \varphi_2 && \text{if } \tau, i \models \varphi_1 \text{ or } \tau, i \models \varphi_2 \\
\tau, i &\models \mathsf{X}\varphi && \text{if } i+1 \leq len(\tau) \text{ and } \tau, i+1 \models \varphi \\
\tau, i &\models \varphi_1 \ \mathsf{U} \ \varphi_2 && \text{if } \tau, j \models \varphi_2 \text{ for some } j \text{ s.t. } i \leq j \leq len(\tau) \\
& && \text{and } \tau, k \models \varphi_1 \text{ for every } k \text{ s.t. } i \leq k < j
\end{aligned}$$

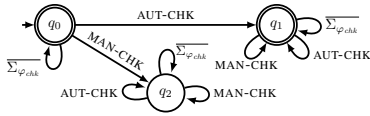We say that τ satisfies φ, written $\tau \models \varphi$, if $\tau, 1 \models \varphi$.

Figure 1: Graphical representation of the DFA for the LTL_p formula $\varphi_{chk} = (\neg\text{MAN-CHK}) \ \mathsf{U} \ \text{AUT-CHK}$. Each transition labelled $\overline{\Sigma_{\varphi_{chk}}}$ is a placeholder for the set of transitions connecting the same pair of states, one per activity in the set $\overline{\Sigma_{\varphi_{chk}}}$ (i.e., different from MAN-CHK and AUT-CHK).

The other boolean connectives *true*, *false*, $\wedge$, $\rightarrow$ are derived as usual, noting $true = \bigvee_{a_i \in \Sigma} a_i$. Further key temporal operators are derived from X and U. E.g., $\mathsf{F}\varphi = true \ \mathsf{U} \ \varphi$ states that φ is *eventually* true in some future instant.

Since every LTL_p formula is an LTL_f formula, the automata-theoretic machinery defined for LTL_f (De Giacomo and Vardi 2013; De Giacomo et al. 2022) applies to LTL_p as well. Specifically, we recall the definition of a deterministic finite-state automaton (DFA) over process traces: a standard DFA with the only difference that, due to how process traces are defined, it employs Σ instead of 2^{Σ} in labelling its transitions. a DFA over process traces from Σ is a tuple $A = \langle \Sigma, Q, q_0, \delta, F \rangle$, where: *(i)* Q is a finite set of states; *(ii)* $q_0 \in Q$ is the initial state; *(iii)* $\delta : Q \times \Sigma \rightarrow Q$ is the (Σ-labelled) transition function; *(iv)* $F \subseteq Q$ is the set of final states. A process trace $\tau = a_1, \ldots, a_n$ is accepted by A if there is a sequence of $n+1$ states $q_0, \ldots, q_n$ such that: *(i)* the sequence starts from the initial state q_0 of A; *(ii)* the sequence culminates in a last state, that is, $q_n \in F$; *(iii)* for every $i \in \{1, \ldots, n\}$, we have $\delta(q_{i-1}, a_i) = q_i$. Notably, every LTL_p formula φ can be encoded into a DFA over process traces A_φ that recognises all and only those traces that satisfy φ: for every process trace τ over Σ, we have that τ is accepted by A_φ if and only if $\tau \models \varphi$.

Given an LTL_p formula φ, we denote by Σ_φ the set of activities mentioned in φ, and by $\overline{\Sigma_\varphi}$ the set of *other* activities, that is, activities not mentioned in φ: $\overline{\Sigma_\varphi} = \Sigma \setminus \Sigma_\varphi$.

Figure 1 shows the DFA of LTL_p formula φ_{chk} from §2, which states that during negotiations, an AUT-CHK must be eventually done, and MAN-CHK, if done, can only follow the automated one.

3.2 Counterfactual Explanations

Differently from other XAI methods, counterfactual explanations do not attempt to explain the inner logic of a predictive model, but rather offer alternatives to the user to obtain a desired prediction (Wachter, Mittelstadt, and Russell 2017).

Given a black-box classifier h_θ, a counterfactual τ_c of τ is a sample for which the prediction of the model is different from the one of τ (i.e., $h_\theta(\tau_c) \neq h_\theta(\tau)$), such that the difference between τ and τ_c is *minimal* (Guidotti 2022). A counterfactual explainer is a function F_t, where t is the number of requested counterfactuals, such that, for a given sample τ, a black box model h_θ, and the set $\mathcal{T}$ of samples used to train the black-box model, returns a set $\mathcal{C} = \{\tau_{c_1}, \ldots, \tau_{c_h}\}$ (with $h \leq t$). For instance, from the running example, with $t = 5$, $y = \textbf{Fail}$, $y' = \textbf{Accept}$, and $\tau = \tau_1$, running $F_t(\tau, h_\theta, \mathcal{T})$

yields the counterfactuals $\tau_{c_1}, \ldots, \tau_{c_5}$.

The XAI literature outlines several desiderata for counterfactual explanations (Verma, Dickerson, and Hines 2020): (i) *Validity:* Counterfactuals should flip the original prediction, aligning with the desired class. (ii) *Input Closeness:* Counterfactuals should minimize changes for clearer explanations. (iii) *Sparsity:* Counterfactuals should alter as few attributes as possible for conciseness. (iv) *Plausibility:* Counterfactuals must adhere to observed feature correlations, ensuring feasibility and realism. (v) *Diversity:* A set of counterfactuals should provide diverse alternatives for the user.

The *validity* of a counterfactual τ_c (desideratum (i)) is measured by the function *val*, which evaluates the difference between the predicted value $h_\theta(\tau_c)$ and the desired class y':

$$\text{val}(h_\theta(\tau_c), y') = I_{h_\theta(\tau_c) \neq y'} \tag{1}$$

where I is the indicator function.[1]

Input closeness of the τ_c to τ (desideratum (ii)), measured by the *dist* function, assesses their dissimilarity:

$$\text{dist}(\tau, \tau_c) = \frac{1}{len(\tau)} \sum_{i=1}^{len(\tau)} d(\tau(i), \tau_c(i)) \tag{2}$$

where d is a properly defined distance in the feature space. In this work, for the distance d between trace elements, we use the indicator function $d(x, y) = I_{x \neq y}$.

The *sparsity* of τ_c regarding τ (desideratum (iii)) is measured through the *spars* function, which counts the number of changes in the counterfactual:

$$\text{spars}(\tau, \tau_c) = ||\tau - \tau_c||_0 = \sum_{i=1}^{len(\tau)} I_{\tau(i) \neq \tau_c(i)} \tag{3}$$

Implausibility (desideratum (iv)) of τ_c is measured by the *implaus* function, calculating the distance between τ_c and the closest sample τ_z in the reference population $\mathcal{T}$:

$$\text{implaus}(\tau_c, \mathcal{T}) = \min_{\tau_z \in \mathcal{T}} \frac{1}{len(\tau)} \sum_{i=0}^{len(\tau)} d(\tau_z(i), \tau_c(i)) \tag{4}$$

where d is the same distance used in (2).

The fifth desideratum (v), *diversity*, measures the pairwise distances between the counterfactuals in $\mathcal{C}$, using the *dive*:

$$\text{dive}(\mathcal{C}) = \frac{1}{|\mathcal{C}|(|\mathcal{C}| - 1)} \sum_{\{\tau_c, \tau_{c'}\}} \text{dist}(\tau_c, \tau_{c'}) \tag{5}$$

where *dist* is defined in (2) and the sum is over all possible unordered pairs $\{\tau_c, \tau_{c'}\}$ of elements $\tau_c, \tau_{c'}$ of $\mathcal{C}$.

3.3 Genetic Algorithms

Genetic Algorithms (GAs) are powerful optimisation techniques, inspired from the natural processes of evolution, widely used for complex problems due to their effectiveness (Mitchell 1998). GAs work with a population of solutions $\mathcal{P}$, evaluating their quality through a *fitness function*. We will present the main components of GAs through the

---

[1] The indicator function $I_{x \neq y}$ is 1 when $x \neq y$ and 0 otherwise.

prism of traces and counterfactual explanations. Each candidate solution in the search space, such as each possible trace, is described by a set of genes, forming its *chromosome* or *genotype*. Below, we outline the main components of a GA.

The *chromosome* τ_c of a candidate solution is a sequence of genes, where $\tau_c(i)$ refers to the i-th gene, $i \in \{1, \ldots, n\}$ where n is the length of the chromosome. In the case of traces $\tau_c(i)$ represents the i-th executed activity.

The *fitness function* $f(\tau_c)$ evaluates the quality of each candidate solution $\tau_c \in \mathcal{P}$, providing a measure that is used as the objective for the GA optimization. For example, in counterfactual generation, the fitness function instantiates the different objectives introduced in Sect. 3.2.

The *initial population* $\mathcal{P}$ is a set of candidate traces generated at the beginning of the GA process. It may consist of randomly generated or predefined candidates: $\mathcal{P} = \{\tau_{c_1}, \tau_{c_2}, \ldots, \tau_{c_p}\}$ where τ_{c_k} denotes an individual candidate. Afterwards, a subset of the population $\mathcal{P}$ is selected based on the fitness score. Selected chromosomes (parents) undergo *crossover* to produce offsprings. This operation combines parts of two parents' chromosomes to create new ones, promoting genetic diversity. If τ_{p_1} and τ_{p_2} are two parents, then each component of the offspring τ_o is selected from either of the two, i.e. $\tau_o(i) = \tau_{p_1}(i)$ or $\tau_{p_2}(i)$.

Offspring chromosomes are subject to *mutation*, which involves randomly altering one or more genes. For example, a gene $\tau_o(i)$ might change with some small probability p_{mut}, introducing new genetic material and helping to prevent premature convergence. Specifically, $\tau_o'(i)$ is mutated to a random gene with probability p_{mut} and remains equal to $\tau_o(i)$ otherwise. Once this offspring undergoes mutation (τ_o'), it becomes part of the new population as τ_{c_k}.

In our example, $\mathcal{P}$ consists of counterfactual candidates $\mathcal{P} = \{\tau_{c_1}, \ldots, \tau_{c_5}\}$. Each candidate's genotype, e.g., τ_{c_1}, is an activity sequence, with each activity as a gene (e.g., $\tau_{c_1}(1) = \text{APPLY}$). The fitness function evaluates candidates based on factors like the ability to flip the outcome, where the fittest are selected for crossover and mutation. Ensuring compliance with constraints, like maintaining the correct sequence of activities (AUT-CHK before MAN-CHK), is crucial to generating compliant counterfactuals, such as c_2, c_4.

4 Approach

We are now ready to introduce our framework for generating counterfactual explanations that comply with background knowledge described through LTL_p formulae.

The first step of is to define a new *compliance* desideratum (desideratum vi). This is done by using a compliance function measuring whether the counterfactual τ_c satisfies a LTL_p formula φ representing the temporal background knowledge. A counterfactual τ_c is deemed as a *temporally constrained counterfactual* if it satisfies φ. Formally:

$$\text{compliance}(\tau_c, \varphi) = \begin{cases} 1 & \text{if } \tau_c \models \varphi \\ 0 & \text{otherwise.} \end{cases} \quad (6)$$

Recalling $\varphi = \varphi_{chk} \wedge \varphi_{comm}$ and counterfactuals $\tau_{c_1}, \ldots, \tau_{c_5}$ from §2, we have $\text{compliance}(\tau_c, \varphi) = 1$ for $c \in \{c_2, c_4\}$ and $\text{compliance}(\tau_c, \varphi) = 0$ for $c \in \{c_1, c_3, c_5\}$.

Next we formulate the fitness function and its afferent objectives (§4.1), and we introduce the modified crossover and mutation operators for the generation of counterfactuals that guarantee the compliance to LTL_p formulae (§4.2).

4.1 Optimisation Problem Formulations

We follow GA-based methods like (Schleich et al. 2021; Dandl et al. 2020) and instantiate the first four desiderata from §3.2 into corresponding optimisation objectives for the fitness function, including the *compliance* desideratum.[2]

Hence, the objectives to optimise are: *validity* of the counterfactual τ_c (1); the *distance* of τ_c to the original trace τ (2); *sparsity*, quantifying the number of changes in τ_c (3) from τ; *implausibility*, that corresponds to the distance of τ_c from the reference population $\mathcal{T}$ (3); and *compliance*, measuring whether the τ_c is compliant to φ or not (6). The resulting fitness function $\mathbf{f}$ is thus defined as:

$$\begin{aligned} \mathbf{f} = & \text{val}(h_\theta(\tau_c), y') + \alpha \, \text{dist}(\tau, \tau_c) + \beta \, \text{spars}(\tau, \tau_c) + \\ & \gamma \, \text{implaus}(\tau_c, \mathcal{T}) + \delta \, \text{compliance}(\tau_c, \varphi). \end{aligned} \quad (7)$$

where $\alpha, \beta, \gamma, \delta$ are weighting factors controlling the influence of each term on the overall fitness.

4.2 Temporal Knowledge-Aware Operators

To guarantee the satisfaction of the background knowledge in the form of φ, we modify the GA, specifically the crossover and mutation operators introduced in §3.3.

Temporal Knowledge-aware Crossover Given an original query instance (process trace) τ satisfying LTL_p formula φ, and two parent traces τ_{p_1} and τ_{p_2} in the current population $\mathcal{P}$, the Temporal Knowledge-aware Crossover operator presented in Algorithm 1 generates an offspring individual τ_o that satisfies φ. It takes as input also the crossover probability p_c and the alphabet Σ_φ of the activities mentioned in φ. The Temporal Knowledge-aware Crossover operator initiates an offspring individual τ_o by retaining from τ the phenotype that actively interacts with φ, guaranteeing its satisfaction. This is formed by those activities in Σ_φ (lines 3–7). The empty genes in the offspring individual chromosome are then filled with one of the two parents' genetic material, but only if the corresponding parent's gene does not interact with φ, i.e., contains an activity in from $\overline{\Sigma_\varphi}$ (lines 8–16). In detail, a random probability p is sampled (line 9) for every empty gene (line 10). The genetic material is then chosen, as in classical crossover operators, from either parent τ_{p_1} or τ_{p_2} (lines 11–12), according to the given crossover probability p_c, if the parent's activity does not belong to Σ_φ. Otherwise, if both parents' activities at that gene belong to Σ_φ, the crossover operator uses the gene from the original query instance τ (line 13). As we prove later, this ensures that τ_o alters τ in a way that does not affect the satisfaction of φ.

Temporal Knowledge-aware Mutation We constrain the mutation operator, designed to increase the diversity of the population, with two strategies that maintain the diversity in the generated offsprings while ensuring that φ is satisfied.

---

[2]In GAs, diversity is managed through selection, crossover, and mutation operators, rather than the fitness function.

Algorithm 1: Temporal Knowledge-aware Crossover operation

1: **Input:** parent individuals τ_{p_1} and τ_{p_2}, crossover probability p_c, original query instance τ, activities Σ_φ
2: **Output:** offspring trace τ_o
3: **for** i from 1 to $|\tau|$ **do**
4:    **if** $\tau(i) \in \Sigma_\varphi$ **then** $\tau_o(i) \leftarrow \tau(i)$
5:    **else** $\tau_o(i) \leftarrow$ *null*
6:    **end if**
7: **end for**
8: **for** i from 1 to $|\tau_o|$ **do**
9:    $p \sim U(0,1)$
10:    **if** $\tau_o(i)$ is *null* **then**
11:       **if** $p < p_c \wedge \tau_{p_1}(i) \notin \Sigma_\varphi$ **then** $\tau_o(i) \leftarrow \tau_{p_1}(i)$
12:       **else if** $p \geq p_c \wedge \tau_{p_2}(i) \notin \Sigma_\varphi$ **then** $\tau_o(i) \leftarrow \tau_{p_2}(i)$
13:       **else** $\tau_o(i) \leftarrow \tau(i)$
14:       **end if**
15:    **end if**
16: **end for**
17: **return** τ_o

Algorithm 2: Compute Safe Activities

1: **function** SAFEACT(τ_o, i, A)
2:    $\Sigma^{\text{safe}} \leftarrow \{\}$
3:    $q \leftarrow$ initial state of A
4:    **for** j from 1 to $i-1$ **do** $q \leftarrow \delta(q, \tau_o(j))$
5:    **end for**
6:    $q' \leftarrow \delta(q, \tau_o(i))$
7:    **for** $a \in \Sigma$ **do**
8:       **if** $\delta(q, a) = q'$ **then** $\Sigma^{\text{safe}} \leftarrow \Sigma^{\text{safe}} \cup \{a\}$
9:       **end if**
10:    **end for**
11:    **return** Σ^{safe}
12: **end function**

Algorithm 3: Temporal Knowledge-aware Mutation operator

1: **Input:** offspring τ_o, mutation probability p_{mut}, LTL_p formula φ, domains of each gene D, mutation strategy S
2: **Output:** mutated offspring τ_o'
3: **for** i from 1 to $|\tau_o|$ **do**
4:    $p \sim U(0,1)$
5:    **if** $p < p_{\text{mut}}$ **then**
6:       **if** S is `aPriori` **and** $\tau_o(i) \notin \Sigma_\varphi$ **then**
7:          $\tau_o(i) \sim U(\mathcal{D}_i \setminus \Sigma_\varphi)$
8:       **else if** S is `Online` **then**
9:          $\Sigma^{\text{safe}} \leftarrow$ SAFEACT(τ_o, i, A_φ)
10:          $\tau_o(i) \sim U(\mathcal{D}_i \cap \Sigma^{\text{safe}})$
11:       **end if**
12:    **else** $\tau_o(i) \leftarrow \tau_o(i)$
13:    **end if**
14: **end for**
15: $\tau_o' \leftarrow \tau_o$
16: **return** τ_o'

The first strategy, called `aPriori`, computes all the possible mutations for each gene $\tau_o(i)$ at the beginning of the mutation phase. The second strategy, called `Online`, exploits DFA A_φ to compute the possible mutations for the current gene $\tau_o(i)$ in the construction of τ_o taking into account the already constructed partial trace $\tau_o(: i-1)$.

We indicate with $\mathcal{D}_i$ the set of all possible activities that can occur at the i-th gene in any generated counterfactual, and define it as the set of all activities occurring in position i in all historical traces $\mathcal{T}$. Formally, $D_i = \{\tau(i) \mid \tau \in \mathcal{T}\}$. Given an offspring τ_o, the `aPriori` strategy produces an offspring τ_o' by mutating only genes that are not in Σ_φ with values that are not in Σ_φ, thus mapping *other* activities into *other* activities, which interact with φ interchangeably. The `Online` strategy instead produces an offspring τ_o' by computing, for every partial trace $\tau_o(: i)$ with $1 \leq i \leq |\tau_o|$, which activities could be used in place of the last activity $\tau_o(i)$ to alter such position i without changing the satisfaction of φ. This is realized through the SAFEACT function defined in Algorithm 2, exploiting the DFA A_φ as follows. First A_φ is traversed using the sequence of activities in $\tau_o(: i-1)$, leading (deterministically) to a state q of the DFA. Then it is checked which next state q' is obtained by applying transition $\delta(q, \tau_o(i))$. The safe activities that can be used in place of $\tau_o(i)$ in state q are then those that lead to the same next state q'. In a sense, this generalises `aPriori`, as in some states also activities from Σ_φ may be interchangeable.

The Temporal Knowledge-aware Mutation operator, presented in Algorithm 3, focuses on mutating an offspring individual τ_o while preserving the satisfaction of the formula φ. The operator takes as input the offspring τ_o, the mutation probability p_{mut}, the set of the domains of the genes $D = \{\mathcal{D}_i \mid i \in \{1, \ldots, |\tau_o|\}\}$, and the chosen mutation strategy S, returning the mutated offspring τ_o' as output. It also takes as input the LTL_p formula φ capturing background knowledge, for which we assume that the DFA A_φ has been

pre-computed (and hence is passed as implicit parameter to the algorithm). The algorithm starts by sampling a random mutation probability p_{mut} (line 4), and then iterates through each gene from 1 to $|\tau_o|$ (lines 3–14). The mutation is carried on if the sampled probability p_{mut} is under the set threshold probability for mutation p_{mut} (line 5), otherwise, we return the value of $\tau_o(i)$ (line 12). In the case of mutation, the value of $\tau_o(i)$ is then uniformly sampled according to the selected mutation strategy S (lines 6–11), mutating each gene accordingly, and returning the mutated offspring τ_o'.

Correctness of the approach. All in all, the application of the crossover operator from Algorithm 1, as well as that of the mutation operator with the two illustrated strategies from Algorithm 3, are correct in terms of how they interact with temporal background knowledge, in the following sense.

Theorem 1. *Let φ be a LTL_p formula, and τ_{p_1}, τ_{p_2}, and τ be process traces over Σ, with $\tau \models \varphi$. Let $p_c \in \mathbb{R}_{[0,1]}$. Assume that Algorithm 1 is invoked by passing τ_{p_1}, τ_{p_2}, p_c, τ, and Σ_φ as input, and that it returns τ_o. Then $\tau_o \models \varphi$.*

Proof. Let $n = len(\tau)$. Upon inspection of Algorithm 1, one can see that every output τ_o produced by the algorithm relates to the input trace τ as follows:

1. $len(\tau_o) = len(\tau) = n$;
2. for every $i \in \{1, \ldots, n\}$:
   (a) if $\tau(i) \in \Sigma_\varphi$ then $\tau_o(i) = \tau(i)$;
   (b) if instead $\tau \notin \Sigma_\varphi$, that is, $\tau \in \overline{\Sigma_\varphi}$, then $\tau_o(i) \in \overline{\Sigma_\varphi}$ as well – equivalently, $\tau_o(i) \in \overline{\Sigma_\varphi}$ if and only if $\tau(i) \in \overline{\Sigma_\varphi}$.

By absurdum, imagine that $\tau_o \not\models \varphi$. Since, by property (1) above, $len(\tau_o) = len(\tau)$, this means that the violation must occur due to a mismatch in the evaluation of an atomic formula in some instant. Technically, there must exist $i \in \{1, \ldots, n\}$ and an atomic sub-formula $a \in \Sigma$ of φ (which, by definition, requires $a \in \Sigma_\varphi$) such that either:

(A) $\tau, i \models a$ and $\tau_o, i \not\models a$, or

(B) $\tau, i \not\models a$ and $\tau_o, i \models a$.

Case (A). By the LTL$_p$ semantics, we have $\tau(i) = a$ and $\tau_o(i) \neq a$. However, this is impossible: since a belongs to Σ_φ, then by property (2a) above, we have $\tau(i) = \tau_o(i)$.

Case (B). By the LTL$_p$ semantics, $\tau, i \not\models a$ if and only if $\tau(i) = b$ for some $b \in \Sigma \setminus \{a\}$. There are two sub-cases: either $b \in \Sigma_\varphi \setminus \{a\}$, or $b \in \overline{\Sigma_\varphi}$. In the first sub-case, impossibility follows again from the fact that, by property (2a) above, since $b \in \Sigma_\varphi$, then $\tau_o(i) = \tau(i) = b$, which implies $\tau_o, i \not\models a$. In the second sub-case, impossibility follows from the fact that $\tau_o(i)$ cannot be a, since by property (2b), the fact that $\tau(i) \in \overline{\Sigma_b}$ implies that also $\tau_o(i) \in \overline{\Sigma_b}$. $\square$

Theorem 2. *Let φ be a LTL$_p$ formula, τ_o a process trace over Σ s.t. $\tau_o \models \varphi$, $D = \{\mathcal{D}_i\} \subset \Sigma^{|\tau_o|}$ the domains of each gene, and $p_{mut} \in \mathbb{R}_{[0,1]}$. Assume that Algorithm 3 is invoked by passing $\tau_o, \varphi, p_{mut}, D, S$ as input (with $S \in \{\texttt{aPriori}, \texttt{Online}\}$), and that it returns τ_o'. Then $\tau_o' \models \varphi$.*

Proof. Correctness of aPriori is proven analogously of Theorem 1. Correctness of Online derives directly from the correspondence between the traces that satisfy φ, and the traces accepted by the DFA of φ, A_φ. From the definition of DFA acceptance, we have that since trace τ_o satisfies φ, there is a sequence $q_0, \ldots, q_n$ of states of A_φ, such that: *(i)* the sequence starts from the initial state q_0 of A; *(ii)* the sequence culminates in a last state, that is, $q_n \in F$; *(iii)* for every $i \in \{1, \ldots, n\}$, we have $\delta(q_{i-1}, \tau_o(i)) = q_i$.

From Algorithm 3, we have that trace τ_o is mutated through Online by selecting an instant i, and allowing for replacing $\tau_o(i)$ with an activity a returned from Algorithm 2. From Algorithm 2, we know that activity a is returned if the following property holds: $\delta(q_{i-1}, a) = q_i$. This means that the mutated trace τ_o' that replaces $\tau_o(i)$ with a, and maintains the rest identical, is accepted by A_φ, with the same witnessing sequence of states $q_0, \ldots, q_n$. From the correspondence between A_φ and the LTL$_p$ semantics of φ, we thus have $\tau_o' \models \varphi$. $\square$

We illustrate the two proposed strategies with formula φ_{chk} of the running example of §2. Based on the DFA of the formula (Fig. 1), we know that, from the initial state q_0, if AUT-CHK happens, we reach the state q_1, which is a final state, where we can repeat any activity in Σ. In q_0 we can perform any other activity in $\overline{\Sigma_{\varphi_{chk}}}$, i.e., different from AUT-CHK and MAN-CHK, remaining in q_0. However, if from

q_0 we perform MAN-CHK, we reach q_2 which is a dead-end state and thus τ_1 violates φ. Given the trace τ_1 from our running example, consider the mutation of its fourth component $\tau_1(4) = $ PHONE. The two strategies for the mutation operator give the following mutation possibilities: in the aPriori strategy we exclude activities in Σ_φ from the mutation, so we can mutate $\tau_1(4)$ with activities in $\mathcal{D}_4 \setminus \Sigma_\varphi$; in the Online strategy the transition in the DFA associated with $\tau_1(4)$ is $\delta(q_1, \tau_1(4)) = q_1$. Since from q_1 all activities give the same transition, that is, $\delta(q_1, a) = q_1$ for every $a \in \Sigma$, we can mutate $\tau_1(4)$ with the entire $\mathcal{D}_4$.

5 Evaluation

To evaluate the approaches for generating counterfactuals explanations incorporating temporal background knowledge, we focus on answering the following questions:

RQ1 How do the proposed methods compare with a standard genetic algorithm?

RQ2 How do the proposed methods compare with a baseline strategy of "generation and check", which ensures the satisfaction of the temporal background knowledge?

Both questions are assessed in terms of generation time and quality of the generated counterfactuals (see below).

The goal of **RQ1** is to assess whether the enforcement of temporal background knowledge has an impact on the quality of the generated counterfactuals when compared to traditional GA approaches. The goal of **RQ2** is to evaluate the proposed strategy, which integrates GAs and DFA against an iterative combination of GAs and a checker function that generates the counterfactual and then checks compliance in a trial-and-error fashion. While a comparison with a standard GA in terms of compliance is out of scope of this paper, as these algorithms are not built to ensure this property, **RQ1** also enables us to discuss also this aspect.

Baselines, Datasets and Evaluation metrics We introduce two baseline methods. To answer the first research question, we use a standard GA, *Genetic$_\varphi$* (Gen$_\varphi$), which uses the fitness function Eq. (7), and the standard crossover and mutation operators from (Schleich et al. 2021) [3]. To answer to the second research question we employ *Mutate-And-Retry* (MAR) which pairs the GA method with *compliance* through a trial-and-error mechanism: the generated trace is checked against φ, and the mutation is repeated until *compliance*.

Experiments are conducted using three datasets commonly used in Process Mining, with details reported in Table 1: *Claim Management* (Rizzi, Di Francescomarino, and Maggi 2020) is a synthetic dataset pertaining to a claim management process, where accepted claims are labelled as *true* and rejected claims as *false*; *BPIC2012* (van Dongen 2012) and *BPIC2017* (van Dongen 2012) two real-life datasets about a loan application process, where traces with accepted loan offers are labelled as *true*, and declined offers as *false*.

---

[3]For compatibility reasons, the same fitness function is also used in the proposed methods, aPriori and Online, even though for them the value of compliance is always one since the generated offsprings τ_o always satisfy φ.

Dataset	Traces	Avg. Len.	$|\Sigma|$	Used Prefixes
Claim Management	4800	11	16	7, 10, 13, 16
BPIC2012	4685	35	36	20, 25, 30, 35
BPIC2017	31 413	35	26	20, 25, 30, 35

Table 1: Summary of the dataset characteristics.

We conduct experiments with different datasets, trace lengths, as well as with LTL_p formulae with sizes of Σ_φ. The experiments are performed over traces with variable prefix lengths, reported in Table 1, testing how techniques perform on average with varying amounts of information (Teinemaa et al. 2019). To assess the impact of the number of generated counterfactuals, we test different settings by generating 5, 10, 15, and 20 counterfactuals (Guidotti 2022).

To evaluate the counterfactual approaches, we use five metrics from (Buliga et al. 2023): *Distance* (2); lower is better, *Sparsity* (3); lower is better, *Implausibility* (4); lower is better, *Diversity* (5); higher is better, and *Runtime* in seconds; lower is better. These metrics refer to a single τ and are averaged across the set of generated counterfactuals. In our experiments we omit the hit rate of the counterfactual set (i.e., whether $|\mathcal{C}| = t$), as in all our experiments the hit rate was always 100%.

Experimental procedure For each dataset, LTL_p formula φ, and prefix length, we split the data into $70\% - 10\% - 20\%$ into training, validation, and testing, using a chronological order split. A XGBoost model is trained and optimised using hyperparameter optimisation to identify the best model configuration for each dataset, prefix length, and encoding combination. The set $\mathcal{T}$ used to train the XGBoost model is used as input for the counterfactual generation methods. We tested different LTL_p formulae φ, with different coverage percentages of the possible different activities $|\Sigma_\varphi|/|\Sigma|$ [4]. The specific φ formulas for each dataset

---

[4]Concerning the impact of the complexity of a formula on our work there are two distinct aspects to be considered. First, formula complexity impacts on the construction of the DFA, because the size of the DFA is, in the worst-case, doubly exponential in the length of the formula (De Giacomo and Vardi 2013). However, this is orthogonal to our approach, as we consider that the DFA has been already constructed (De Giacomo, De Masellis, and Montali 2014; Fuggitti 2019). In addition, it is well-known that, despite this worst-case complexity, the size of the DFA is often polynomial in the length of the formula, and that such off-the-shelf techniques incorporate several optimizations (De Giacomo and Favorito 2021). Second, it is less meaningful to relate metrics on the syntactic complexities of a formula with performance, as their interaction occurs at the semantic level. In this respect, focusing on coverage reflects the intuition that when a modeller explicitly mentions an activity in a formula, they do so to express constraints on when such an activity must/can be executed (i.e., a modeller would not express formula "true", which has empty formula signature, with the equivalent formula $\bigvee_{a_i \in \Sigma} a_i$, which has the whole alphabet as formula signature). In this sense, using more activities (i.e., having a larger formula signature) correlates with "constraining more", which in turn impacts on performance. This motivates the choice of coverage, and in turn why in our experiments coverage influences performance.

are presented in the code repository linked in the beginning of the paper. Regarding the coefficients in Eq. (7), after testing multiple configurations, the final configuration was set to $\alpha = 0.5, \beta = 0.5, \gamma = 0.5, \delta = 0.5$ to give all objectives the same weight.

Next, 15 instances are sampled from the test set and used for the counterfactual generation, one trace τ at a time, while the counterfactuals are evaluated using the evaluation framework. Experiments were run on a M1 with 16GB RAM. For the GA setting, we initialise the population through a hybrid approach: selecting close points from the reference population or, if unavailable, by randomly generating traces. We set the number of generations to 100, $p_c = 0.5$, $p_{\text{mut}} = 0.2$. In *population selection*, the top 50% of the population, w.r.t. the fitness function, moves to the next generation. Termination occurs at the max generation number or if no significant performance improvement occurs.

We assess differences using statistical tests: we perform Wilcoxon signed-rank tests (Wilcoxon 1945) for pairwise comparisons, with p-values adjusted by the Bonferroni correction (Dunn 1961). Methods are then ranked by performance on each metric, allowing for clear comparison.

6 Results

Table 2 shows the average values of each metric for each method. In parentheses, we indicate the afferent rank of each value, indicating different rankings only for statistically significant differences. The best performing strategy is highlighted in bold. If multiple values are in bold, this suggests no statistically significant difference between the methods in terms of the respective metric (or, in other words, that there is no statistically significant difference between the absolute best of the row and the other bold values). This also directly translates to multiple methods having the same rank.

We show the evolution of the results with three levels of coverage $|\Sigma_\varphi|/|\Sigma|$: 10%, 25%, and 50%.

Answering RQ1. We start by considering the time performance. Looking at the *runtime* of the three methods, we observe that $Genetic_\varphi$ and aPriori perform similarly, with $Genetic_\varphi$ being among the best performers 7 out of 9 times, while aPriori 5 out of 9 times. Online instead shows an increase in the time required for generating the counterfactuals, especially for large coverage rates. Regarding quality, a different story emerges: aPriori and Online demonstrate a good performance, with aPriori showing a consistent ability to generate closer, sparser, and more plausible counterfactuals w.r.t., Gen_φ. This performance is more pronounced in real datasets (BPIC2012, and especially BPIC2017), where the complexity and length of traces are challenging the capability of Gen_φ. On the contrary, in the simpler Claim Management dataset, the superiority of aPriori becomes more apparent only for higher coverage levels, while for lower coverages, Gen_φ exhibits better counterfactual quality results. Concerning Online, it excels in balancing counterfactual quality with diversity, particularly in BPIC2012, making it a viable candidate in scenarios where both aspects are critical.

Cover.	Metric	Claim Management				BPIC2012				BPIC2017			
		Gen$_\varphi$	MAR	aPriori	Online	Gen$_\varphi$	MAR	aPriori	Online	Gen$_\varphi$	MAR	aPriori	Online
10%	Dist.	**0.48** (1)	**0.50** (1)	**0.50** (1)	**0.50** (1)	0.54 (4)	0.50 (2)	**0.47** (1)	0.50 (2)	0.59 (4)	0.50 (2)	**0.44** (1)	0.50 (2)
	Spars.	**2.50** (1)	**2.58** (1)	2.63 (1)	**2.58** (1)	7.69 (4)	**6.95** (1)	6.79 (1)	6.97 (1)	6.95 (4)	**5.60** (1)	**5.20** (1)	5.85 (3)
	Impl.	**8.49** (1)	8.88 (2)	8.84 (2)	8.92 (2)	**7.49** (1)	**7.37** (1)	7.26 (1)	7.38 (1)	**7.51** (1)	**6.81** (1)	6.59 (1)	**6.86** (1)
	Dive	0.48 (4)	**0.54** (1)	**0.54** (1)	**0.54** (1)	0.35 (4)	**0.40** (1)	0.41 (1)	**0.40** (1)	**0.58** (1)	0.49 (2)	0.44 (4)	0.49 (2)
	Runtime	**2.53** (1)	4.65 (4)	4.38 (2)	4.09 (2)	25 (3)	90 (4)	**12** (1)	17 (2)	**29** (1)	78 (2)	57 (2)	65 (2)
25%	Dist.	**0.48** (1)	**0.47** (1)	0.50 (1)	**0.47** (1)	0.47 (2)	0.38 (4)	**0.30** (1)	0.36 (2)	0.58 (4)	0.33 (2)	**0.22** (1)	0.31 (2)
	Spars.	2.50 (1)	2.45 (1)	2.58 (4)	2.42 (1)	7.01 (3)	5.24 (3)	**4.38** (1)	4.92 (2)	6.77 (4)	4.02 (2)	**2.72** (1)	3.82 (2)
	Impl.	**8.46** (1)	8.80 (4)	8.83 (1)	8.70 (1)	**6.38** (1)	9.73 (2)	9.67 (2)	9.70 (2)	7.39 (4)	**5.52** (1)	4.94 (1)	5.40 (1)
	Dive	0.47 (3)	0.50 (2)	0.52 (1)	0.49 (3)	0.36 (2)	**0.38** (1)	0.32 (4)	0.35 (2)	**0.60** (1)	0.34 (2)	0.26 (4)	0.33 (2)
	Runtime	**2.76** (1)	4.41 (4)	3.10 (1)	3.23 (1)	154 (1)	211 (1)	133 (1)	191 (1)	29.1 (1)	81.6 (3)	54.4 (2)	75.6 (3)
50%	Dist.	0.41 (4)	0.27 (2)	**0.22** (1)	0.27 (2)	0.55 (4)	0.20 (3)	**0.24** (1)	0.30 (2)	0.6 (4)	0.17 (2)	**0.09** (1)	0.18 (2)
	Spars.	2.19 (4)	1.41 (2)	**1.17** (1)	1.40 (2)	7.83 (4)	5.55 (3)	**3.52** (1)	4.11 (2)	7.11 (4)	2.06 (2)	**1.06** (1)	2.16 (2)
	Impl.	7.17 (2)	7.17 (2)	**7.01** (1)	7.16 (2)	**7.58** (1)	8.51 (4)	7.28 (2)	**7.13** (1)	7.95 (4)	4.86 (2)	**4.38** (1)	5.14 (2)
	Dive	**0.43** (1)	0.30 (2)	0.25 (4)	0.30 (2)	**0.36** (1)	0.19 (4)	0.25 (2)	0.27 (2)	**0.61** (1)	0.18 (2)	0.11 (4)	0.19 (2)
	Runtime	**3.39** (1)	9.78 (4)	4.80 (2)	5.18 (2)	535 (2)	1500 (4)	**451** (1)	780 (2)	**288** (1)	566 (3)	**261** (1)	463 (3)

Table 2: Performance metrics across different datasets. The ranking position of each method is indicated in parentheses.

A final remark on the *compliance* of the counterfactuals generated with *Genetic$_\varphi$*. Assessing *Genetic$_\varphi$*'s *compliance* is methodologically complex and beyond this paper's scope, as it is not designed to ensure compliance, which may depend upon several factors, including the LTL$_p$ formulae used. Nonetheless, it is worth noting that in our experiments *Genetic$_\varphi$* managed to reach a compliance ranging on average from 80% to 99%, depending upon the coverage used. This hints that *Genetic$_\varphi$* can achieve good overall *compliance* but cannot always guarantee satisfaction of φ.

Answering RQ2. Overall, the *runtime* performance analysis reveals that `aPriori` and `Online` demonstrate significantly low *runtime* across all datasets and coverage levels, enabling a quick counterfactual generation. This efficiency is particularly pronounced at higher coverage levels, where `aPriori` and `Online` maintain fast processing times, thanks to their optimised mutation processes and reduced need for extensive checks. In contrast, *Mutate-And-Retry* exhibits substantially higher runtimes, especially notable at 50% coverage, due to the complexity of performing the trace validation after each mutation.

In terms of quality of the generated counterfactuals, `MAR` performs similarly to our methods, particularly when the task coverage is lower. As complexity increases, particularly at 50% task coverage, `aPriori` and `Online` outperform `MAR`. The counterfactuals generated by `MAR` tend to have higher *sparsity*, requiring more modifications to the original trace. In contrast, `aPriori` and `Online` are better at generating counterfactuals with less *sparsity*. `MAR` shows good performance in *implausibility*, often ranking close to or at the top. However, `aPriori` and `Online` still maintain a good level of *implausibility* performance, indicating their ability to generate plausible counterfactuals as well, but with added efficiency and sparsity. Finally, `MAR` does not show a significant improvement in *diversity*, compared to our two strategies. Thus, `aPriori` and `Online` offer a good trade-off between *runtime* and quality of the generated counterfactuals, which remains overall comparable to that of `MAR`.

7 Related Work

In this section, we review related work on counterfactual explanations, focusing on their use in Predictive Process Monitoring (PPM) and temporal data. We begin by examining general techniques for generating counterfactuals, distinguishing between case-based and generative approaches. We then explore how Genetic Algorithms (GAs) are employed in counterfactual generation, noting their advantages and limitations. Lastly, we dive into recent developments in counterfactual explanations for temporal process data.

Counterfactual explanations in XAI Counterfactual explanations identify minimal changes to alter a model's prediction (Wachter, Mittelstadt, and Russell 2017).Techniques for generating counterfactuals fall into two categories: case-based, which find counterfactuals within the sample population, and generative, creating them through optimisation-based techniques (e.g., hill-climbing algorithms) (Verma, Dickerson, and Hines 2020).

Genetic Algorithms (GAs) are widely used for generating counterfactuals by optimizing a population of potential candidates through a fitness function (Dandl et al. 2020; Schleich et al. 2021). Both single-objective and multi-objective GA solutions are available, with single-objective solutions converging faster due to lower complexity (Schleich et al. 2021) and multi-objective GAs providing multiple optimal solutions using a Pareto Front (Dandl et al. 2020). One key benefit of GAs is their ability to maximize population diversity, yet their stochastic nature often leads to inconsistent results. Moreover, unlike gradient-based optimisation techniques for counterfactual generation (Mothilal, Sharma, and Tan 2020), which require the use of differentiable models to compute counterfactuals, GAs do not require access to the model's parameters or gradient computation. As such, they are not limited to differentiable models and do not require gradient computations by construction.

Despite the advancements in the literature that try to incorporate plausibility and causality constraints (Schleich et al. 2021), current methods have limitations in ensuring the feasibility of counterfactuals. Feasibility is crucial for generating valid counterfactuals, typically enforced through restricting the data manifold, specifying constraints, or mini-

mizing distances to training set points (Maragno et al. 2024).

As mentioned by (Beckh et al. 2023), no background knowledge injection has been explored so far for the generation of counterfactual explanations, which is a challenge also when focusing on counterfactual explanations generated with GA approaches (Zhou and Hu 2024).

Counterfactual explanations for temporal data Four works so far have tackled the counterfactual explanation problem in the PPM domain (Huang, Metzger, and Pohl 2022; Hsieh, Moreira, and Ouyang 2021; Hundogan et al. 2023; Buliga et al. 2023).

The first paper introduces LORELEY, an adaptation of the Local Rule-Based Explanations (LORE) framework (Guidotti et al. 2019), which generates counterfactual explanations leveraging a surrogate decision tree model using a genetically generated neighbourhood of artificial data instances to be trained (Huang, Metzger, and Pohl 2022; Guidotti et al. 2019). The prediction task the authors address is the one of multi-class outcome prediction. To ensure the generation of feasible counterfactuals, LORELEY imposes process constraints in the counterfactual generation process by using the whole prefix of activities as a single feature, encoding the whole control-flow execution as a variant of the process.

The second work presents DICE for Event Logs (DICE4EL) (Hsieh, Moreira, and Ouyang 2021). DICE4EL extends one of the methods found within DICE (Mothilal, Sharma, and Tan 2020), specifically, the gradient-based optimisation method by adding a feasibility term to ensure that the generated counterfactuals maximise the likelihood of belonging to the training set. To do so, DICE4EL leverages a Long-Short Term Memory (LSTM)-based predictive model as it requires gradients for the counterfactual explanation search. The prediction task addressed in the paper is that of next activity prediction with a milestone-aware focus.

The third, the most recent approach for generating counterfactual explanations for PPM, CREATED, leverages a genetic algorithm to generate candidate counterfactual sequences (Hundogan et al. 2023). To ensure the feasibility of the data, the authors build a Markov Chain, where each event is a state. Then, using the transition probabilities from one state to another, they can determine how likely a counterfactual is, given the product of the states.

The fourth and final work looked into proposing an evaluation framework for measuring counterfactual explanations in PPM by proposing a novel metric measuring the conformance of counterfactual generation techniques (Buliga et al. 2023). As noted by the authors, no previous approaches make use of temporal background knowledge explicitly when generating counterfactual explanations. However, background knowledge can play an important role in ensuring the feasibility of the generated counterfactuals, especially from the perspective of sequences of activities, where different constraints may have a different impact on the outcome of a trace execution. The present work aims to specifically fill this gap identified in the literature.

8 Conclusions

We have introduced a novel framework for generating counterfactual traces in temporal domains, guaranteeing that they respect background knowledge captured in a suitable temporal logic. Our approach blends automata-theoretic techniques of this logic with genetic algorithms. The results of the evaluation show that the strategies we propose ensure that background formulae remain satisfied by the generated counterfactual traces, while these traces also maintain or improve general counterfactual explanation desiderata compared to state-of-the-art methods.

In the future, we aim to develop more efficient genetic operators strategies. We also plan to extend our approach to richer temporal logics dealing not only with activities, but also with numerical data, as in (Felli et al. 2023). This appears viable given the basis provided here, in the light of the automata-theoretic characterisation of such logics (Felli et al. 2023), as well as the fact that counterfactual desiderata can be seamlessly redefined over numerical data.

Acknowledgments

References

Beckh, K.; Müller, S.; Jakobs, M.; Toborek, V.; Tan, H.; Fischer, R.; Welke, P.; Houben, S.; and von Rueden, L. 2023. Harnessing Prior Knowledge for Explainable Machine Learning: An Overview. In *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, 450–463.

Buliga, A.; Di Francescomarino, C.; Ghidini, C.; and Maggi, F. M. 2023. Counterfactuals and Ways to Build Them: Evaluating Approaches in Predictive Process Monitoring. In *International Conference on Adv. Inf. Sys. Eng.*, 558–574. Springer.

Dandl, S.; Molnar, C.; Binder, M.; and Bischl, B. 2020. Multi-Objective Counterfactual Explanations. In Bäck, T.; Preuss, M.; Deutz, A.; Wang, H.; Doerr, C.; Emmerich, M.; and Trautmann, H., eds., *Parallel Problem Solving from Nature – PPSN XVI*, 448–469. Springer.

De Giacomo, G.; De Masellis, R.; Maggi, F. M.; and Montali, M. 2022. Monitoring Constraints and Metaconstraints with Temporal Logics on Finite Traces. *ACM Trans. Softw. Eng. Methodol.*, 31(4).

De Giacomo, G.; De Masellis, R.; and Montali, M. 2014. Reasoning on LTL on Finite Traces: Insensitivity to Infiniteness. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI)*, 1027–1033. AAAI Press.

De Giacomo, G.; and Favorito, M. 2021. Compositional Approach to Translate LTLf/LDLf into Deterministic Finite Automata. *Proceedings of the International Conference on Automated Planning and Scheduling*, 31(1): 122–130.

De Giacomo, G.; and Vardi, M. Y. 2013. Linear Temporal Logic and Linear Dynamic Logic on Finite Traces. In Rossi, F., ed., *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, 854–860. IJCAI/AAAI.

Di Francescomarino, C.; and Ghidini, C. 2022. Predictive Process Monitoring. In *Process Mining Handbook*, volume 448 of *LNBIP*, 320–346. Springer.

Dunn, O. J. 1961. Multiple Comparisons Among Means. *Journal of the American Statistical Association*, 56(293): 52–64.

Felli, P.; Montali, M.; Patrizi, F.; and Winkler, S. 2023. Monitoring Arithmetic Temporal Properties on Finite Traces. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI)*, 6346–6354. AAAI Press.

Fionda, V.; and Greco, G. 2018. LTL on Finite and Process Traces: Complexity Results and a Practical Reasoner. *J. Artif. Intell. Res.*, 63: 557–623.

Fuggitti, F. 2019. LTLf2DFA (Version 1.0.3).

Guidotti, R. 2022. Counterfactual explanations and how to find them: literature review and benchmarking. *Data Mining and Knowledge Discovery*.

Guidotti, R.; Monreale, A.; Giannotti, F.; Pedreschi, D.; Ruggieri, S.; and Turini, F. 2019. Factual and Counterfactual Explanations for Black Box Decision Making. *IEEE Intelligent Systems*, 34: 14–23.

Hsieh, C.; Moreira, C.; and Ouyang, C. 2021. DiCE4EL: Interpreting Process Predictions using a Milestone-Aware Counterfactual Approach. In *ICPM*, 88–95.

Huang, T.-H.; Metzger, A.; and Pohl, K. 2022. Counterfactual Explanations for Predictive Business Process Monitoring. In Themistocleous, M.; and Papadaki, M., eds., *Information Systems*, 399–413. Cham: Springer International Publishing.

Hundogan, O.; Lu, X.; Du, Y.; and Reijers, H. A. 2023. CREATED: Generating Viable Counterfactual Sequences for Predictive Process Analytics. In Indulska, M.; Reinhartz-Berger, I.; Cetina, C.; and Pastor, O., eds., *International Conference on Adv. Inf. Sys. Eng.*, 541–557. Cham: Springer Nature Switzerland.

Maragno, D.; Kurtz, J.; Röber, T. E.; Goedhart, R.; Birbil, S. I.; and den Hertog, D. 2024. Finding Regions of Counterfactual Explanations via Robust Optimization. *INFORMS J. Comput.*, 36(5): 1316–1334.

Mitchell, M. 1998. *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press.

Mothilal, R. K.; Sharma, A.; and Tan, C. 2020. Explaining machine learning classifiers through diverse counterfactual explanations. In Hildebrandt, M.; Castillo, C.; Celis, L. E.; Ruggieri, S.; Taylor, L.; and Zanfir-Fortuna, G., eds., *FAT* *'20: Conference on Fairness, Accountability, and Transparency, Barcelona, Spain, January 27-30, 2020*, 607–617. ACM.

Pnueli, A. 1977. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FOCS)*, 46–57. IEEE.

Rizzi, W.; Di Francescomarino, C.; and Maggi, F. M. 2020. Explainability in predictive process monitoring: When understanding helps improving. In *International Conference on Business Process Management*, 141–158. Springer.

Schleich, M.; Geng, Z.; Zhang, Y.; and Suciu, D. 2021. GeCo: Quality Counterfactual Explanations in Real Time. *Proc. VLDB Endow.*, 14(9): 1681–1693.

Teinemaa, I.; Dumas, M.; La Rosa, M.; and Maggi, F. M. 2019. Outcome-Oriented Predictive Process Monitoring: Review and Benchmark. *ACM Trans. Knowl. Discov. Data*, 13(2).

van der Aalst, W. M. P. 2016. *Process Mining - Data Science in Action, Second Edition*. Springer.

van Dongen, B. 2012. BPI Challenge 2012.

Verma, S.; Dickerson, J. P.; and Hines, K. 2020. Counterfactual Explanations for Machine Learning: A Review. *CoRR*, abs/2010.10596.

Wachter, S.; Mittelstadt, B. D.; and Russell, C. 2017. Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR. *CoRR*, abs/1711.00399.

Wilcoxon, F. 1945. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6): 80–83.

Zhou, R.; and Hu, T. 2024. Evolutionary Approaches to Explainable Machine Learning. In Banzhaf, W.; Machado, P.; and Zhang, M., eds., *Handbook of Evolutionary Machine Learning*, 487–506. Singapore: Springer Nature Singapore.