Leveraging Counterfactuals for Prescriptive Process Analytics

Ngoc-Diem Le, Alessandro Padella, Francesco Vinci, and Massimiliano de Leoni

University of Padua, Padua, Italy
ngocdiem.le@phd.unipd.it, alessandro.padella@unipd.it,
francesco.vinci.1@phd.unipd.it, deleoni@math.unipd.it

Abstract. Prescriptive process analytics aims to provide actionable recommendations for process instances that are predicted to fall short of achieving satisfactory outcomes. A common type of recommendation typically focuses on assigning activities to specific resources, as it is a general task that naturally applies across many domains. Given that processes may involve hundreds of resources, brute-force approaches for evaluating all possible activity-resource combinations are computationally infeasible. Current state-of-the-art techniques, conversely, adopt a sequential approach that selects the most suitable activity and then allocates it to one of the suitable resources: this is inherently sub-optimal. This paper leverages counterfactual generation techniques to formulate recommendations. Counterfactual-based methods offer innovative strategies that efficiently converge to highly effective interventions. Experimental evaluations conducted on several real-life case studies demonstrate that our counterfactual-based technique outperforms a baseline approach that follows a sequential activity-to-resource assignment strategy.

Keywords: Process Mining · Resource Allocation · Process-aware Recommendation Systems · Counterfactual Generation · Process Improvement.

1 Introduction

Process mining aims to discover and improve processes based on the analysis of process transactional data that records how single executions have been carried out. Within this field, prescriptive process analytics focuses on recommending actions to mitigate "risky" executions that are predicted to lead to unsatisfactory outcomes, such as high costs or poor customer satisfaction. One of the most common types of recommendations is the real-time suggestion of resources to carry out certain activities [19]. Prescriptive process analytics is currently gaining momentum, with several techniques having been proposed over the years (see Section 2). This paper contributes by providing recommendations to optimize process performance, with a particular focus on minimizing the total execution time, aligning with a large share of the related work in the field.

Existing techniques typically leverage a divide-and-conquer approach: they first select the activity based on its potential to enhance the process efficiency, and then assign it to a suitable resource [19]. While intuitive, this strategy can result in suboptimal outcomes. Conversely, optimizing the activity-resource pair by evaluating all possible combinations is often intractable in practice, especially in organizations with hundreds of resources.

This paper proposes a technique based on counterfactuals [14,29] to recommend the activity-resource pairs for each running process instance. We use the term *counterfactual* in the context of explainable AI as introduced by [30]: a counterfactual identifies the minimal changes to an input instance that would alter the prediction of the oracle function toward a desired outcome. This extensive employment of counterfactual-based approaches is linked to their nature of defining *what-if* scenarios, such as "*You were denied a loan because your loan duration was 20 years. If your loan duration had been 30 years, you would have been offered a loan*" [30]. This capability makes counterfactual-generation frameworks well-suited for recommending activity-resource pairs. The advantage of prescriptive process analytics is evident: frameworks for counterfactual generation are readily available and can be directly used to generate activity-resource recommendations along with explanations. Also, these frameworks can quickly generate multiple counterfactuals, which translates in our setting to producing several alternative recommendations for the same process instance. Notably, while counterfactuals have been used to explain predictions and recommendations, they have never been used to generate recommendations themselves. This flexibility is beneficial because a rigid resource-to-activity imposition is against the principles of resource-aware recommender systems [9], which emphasize the need to support decision-makers with multiple viable options rather than prescribing fixed assignments.

Our counterfactual-based technique for prescriptive process analytics has been evaluated on several real-life processes and event logs, and compared with a baseline where the best activity is first chosen based on the best predicted next activity, and then it is allocated to one random resource. This baseline is reasonable because, while the number of activities in the process is relatively limited, the number of process resources can be very high, even reaching hundreds (cf. the case studies used in the experiments reported in Section 5 and summarized in Table 1). It is indeed intractable to compute the prediction for each of the hundreds of different resources separately. The evaluation results show that the average potential reduction in process-instance total duration is considerably higher than what the baseline can achieve. This is due to the use of counterfactual generation, which can efficiently build effective recommendations in the form of activity-resource pairs.

The remainder of this paper is organized as follows: Section 2 discusses related works in the domain of prescriptive process analytics and the use of counterfactuals in these fields. Section 3 introduces the necessary background concepts, while Section 4 puts forward our framework that leverages counterfactuals. Section 5 reports on the evaluation setup and results, while discussing limitations in the evaluation. Finally, Section 6 concludes the paper, summarizing the contribution of our paper.

2 Related Works

Prescriptive process analytics has recently gained significant attention, with various methods being applied to provide Key Performance Indicator (KPI)-driven recommendations. Numerous data-driven frameworks have been proposed to recommend the optimal next activity for KPI optimization [12], incorporating AI-based approaches such as Reinforcement Learning [6], Generative Adversarial Networks [20], Linear Temporal Logic formulas [10], and explainability techniques [25]. Adopting a different perspec-

tive, different authors address the problem of correctly timing the interventions [5], aiming to balance the risks of intervening too early (leading to unnecessary delays/costs) or too late (diminishing effectiveness), also guaranteeing a white-box approach [27]. Beyond next-activity recommendations, researchers have demonstrated that KPI improvements can also result from enhanced resource allocation [3, 24, 26].

Literature proposes a few counterfactual frameworks [4, 8, 15–17, 22] to explain the predictions, namely the predictive-monitoring outcome. The work by Bhattacharya et al. [4] is particularly noteworthy. They propose a counterfactual-based visual framework designed to support patients and healthcare professionals in making decisions aimed at improving patient outcomes. However, this framework is specifically tailored to the healthcare domain. More critically, it is not process-centric: while it assists in identifying potential next actions, it does not directly generate recommendations, intended as pairs of suggested activity and performing resource. The same limitation also holds for Mothilal et al. [22]. Furthermore, the effectiveness of the approach by Bhattacharya et al. [4] is evaluated through subjective user feedback, rather than through objective metrics that are measured on real(istic) process executions to assess the actual process improvements.

This paper is the first to introduce a recommendation framework that utilizes counterfactuals to explicitly suggest the next activities to be executed for various ongoing process instances, as well as the resources that should carry them out. It is important to emphasize that our problem setting differs fundamentally from job-shop scheduling, where some counterfactual-based approaches have been proposed (see, e.g., [21]). In job-shop scheduling, the sequence of activities is predetermined and cannot be altered, whereas our approach allows for flexibility in recommending alternative next activities.

3 Preliminaries

The starting point for any process mining technique is an *event log*. An event log is a collection of *traces*, where a trace is a sequence of events with their attributes. Each trace describes the life-cycle of a particular *process instance* (i.e., a *case*) regarding the *activities* executed by certain *resources* at specific *timestamps* and the process *attributes* manipulated.

Definition 1 (Events). *Let $\mathcal{A}$ be the set of process activities. Let $\mathcal{T} \subset \mathbb{N}$ be the set of possible timestamps. Let $\mathcal{R}$ be the set of possible resources. Let $\mathcal{V}$ be the set of process attributes. Let $\mathcal{W}_{\mathcal{V}}$ be a function that assigns a domain $\mathcal{W}_{\mathcal{V}}(x)$ to each process attribute $x \in \mathcal{V}$. Let $\overline{\mathcal{W}} = \cup_{x \in \mathcal{V}} \mathcal{W}_{\mathcal{V}}(x)$. An event is a tuple $(a, t, r, v) \in \mathcal{A} \times \mathcal{T} \times \mathcal{R} \times (\mathcal{V} \not\to \overline{\mathcal{W}})$ where a is the event activity, t is the timestamp associated with the event, r is the resource that performs it, and v is a partial function assigning values to process attributes with $v(x) \in \mathcal{W}_{\mathcal{V}}(x)$.*

Definition 2 (Traces & Event Logs). *Let $\mathcal{E} = \mathcal{A} \times \mathcal{T} \times \mathcal{R} \times (\mathcal{V} \not\to \overline{\mathcal{W}})$ be the universe of events. Let a trace σ be a sequence of events, i.e., $\sigma \in \mathcal{E}^*$. An event-log $\mathcal{L}$ is defined as a set of traces, i.e., $\mathcal{L} \subset \mathbb{B}(\mathcal{E}^*)$.*

Without loss of generality, the events in each trace are considered sorted chronologically. Given an event $e = (a, t, r, v)$, the remainder uses the following shortcuts:

$activity(e) = a$, $time(e) = t$, $resource(e) = r$ and $variable(e) = v$. Furthermore, given a trace $\sigma = \langle e_1, \ldots, e_n \rangle$, $prefix(\sigma)$ denotes the set of all prefixes of σ, including σ, namely $prefix(\sigma) = \{\langle\rangle, \langle e_1 \rangle, \langle e_1, e_2 \rangle, \ldots, \langle e_1, \ldots, e_n \rangle\}$.

In developing our technique, it is imperative to delineate the recommendation objective. We focus on minimizing the total execution time of a trace, as this is a general goal that naturally fits many real-world settings where process efficiency is a priority. The total execution time can be defined as a function over a given trace $\sigma = \langle e_1, \ldots, e_n \rangle \in \mathcal{E}^*$ that returns the difference between the first and the last timestamp related to σ. Note that the total time is assumed to be computed a posteriori when the execution is completed and leaves a complete trail of the timestamps for a particular trace σ. We can now define the total time prediction problem as follows:

Let $\sigma' = \langle e_1, \ldots, e_k \rangle \in \mathcal{L}$ be a prefix of a running case, which eventually will complete as $\sigma = \langle e_1, \ldots, e_k, e_{k+1}, \ldots, e_n \rangle$. The total execution time prediction problem can be formulated as forecasting the value of $\mathcal{T}_{total}(\sigma') = time(e_n) - time(e_1)$. In the process mining literature, this problem has been faced with different models [13,18,23]. We approach the model by estimating a **Total Time Oracle function** $\Phi : \mathcal{E}^* \times \mathcal{A} \times \mathcal{R} \rightarrow \mathbb{N}$, which, for an incomplete trace σ', an activity a and a resource r, forecasts the total execution time $\mathcal{T}_{total}$ if the next event e_{k+1} satisfies the conditions: $activity(e_{k+1}) = a$ and $resource(e_{k+1}) = r$.

Each prediction technique requires the definition of the domain $X_1 \times \ldots \times X_m$ and a **Trace-to-instance Encoding function** $\rho : \mathcal{E}^* \rightarrow X_1 \times \ldots \times X_m$, which maps each (prefix of a) trace σ in a vector $\rho(\sigma) \in X_1 \times \ldots \times X_m$ of m elements that can be of different nature, such as a process attribute, a timestamp, or the number of executions of an activity in σ.

This paper aims to provide a framework that recommends the most appropriate activity-resource pairs for reducing the execution time of running instances. However, to ensure domain validity, we assume that an activity is considered valid in a given process state, which refers to the current execution context defined by the sequence of past events, if it has been observed in similar past executions. This requires providing a state-representation function, as we will explain in more detail below.

Definition 3 (State-representation Function). *Let σ be a trace, and $\mathcal{S}$ a set of possible state representations, where each representation corresponds to a unique state of the process based on the observed events. The function $l^{state} : \mathcal{E}^* \rightarrow \mathcal{S}$ is called the state-representation function, where it returns the state for each (prefix of a) trace.*

Determining the activities allowed after a sequence of events requires building a transition system where nodes represent the observed states, and arcs represent transitions between states. Each arc corresponds to an event and is labeled with the name of the associated activity [1]. Multiple arcs can bear the same label.

Definition 4 (Transition System). *Let l^{state} be a state-representation function, $\mathcal{L}$ an event log, and $\mathcal{E}$ its set of events over a set $\mathcal{A}$ of activities. A transition system abstracting $\mathcal{L}$ is a tuple $\mathcal{TS}_{\mathcal{L}} = (S, T) \subseteq \mathcal{S} \times (\mathcal{S} \times \mathcal{A} \times \mathcal{S})$ where*

- $S = \cup_{\sigma \in \mathcal{L}} \cup_{\sigma' \in prefix(\sigma)} l^{state}(\sigma')$
- $T = \cup_{\sigma \in \mathcal{L}} \cup_{\sigma' \oplus \langle e \rangle \in prefix(\sigma)} (l^{state}(\sigma'), activity(e), l^{state}(\sigma' \oplus \langle e \rangle)$

Figure 1 illustrates an example of a transition system in accordance with Definition 4. It has been built on an event log $\mathcal{L}^{ex} = \{\langle a,b,c,d\rangle, \langle a,c,e\rangle, \langle a,e,b,d\rangle, \langle a,b,d,e\rangle\}$[1] using a sequence-based state-representation $l^{state}_{sequence\_based}(\langle e_1,\ldots,e_n\rangle) = \langle activity(e_1),\ldots,activity(e_n)\rangle$. Through this function, the state of a (prefix of a) trace is identified with its ordered list of activities. In the example with $\mathcal{L}^{ex}$, the set of possible states is thus $\mathcal{S} = \{\langle a,b,c,d\rangle, \langle a,b,d,e\rangle, \langle a,e,b,d\rangle, \langle a,c,e\rangle, \langle a,e,b\rangle, \langle a,b,d\rangle, \langle a,b,c\rangle, \langle a,c\rangle, \langle a,e\rangle, \langle a,b\rangle, \langle a\rangle\}$. Transition systems are built based on historical data, representing which activities are permissible following a given sequence of activities. While these systems can become exceedingly large and unintelligible, this is not a concern as they are used internally and are not exposed to the actors involved in the process.

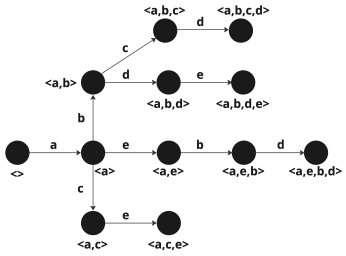


Fig. 1: Transition system for an example log.

The purpose of this work, however, is not to recommend the optimal next activity but rather to identify the most suitable activity-resource pair, ensuring feasibility from both activity and resource perspectives. This requires verifying if a certain resource is eligible to perform a given activity, according to the company and process constraints. The determination of whether a resource r can perform an activity a is based on the information in the event log $\mathcal{L}$; namely, there must exist an event e in the event log $\mathcal{L}$ such that $activity(e) = a$ and $resource(e) = r$. At the end, for each activity $a \in \mathcal{A}_{\sigma'}$, we identify the set of **eligible resources** $\mathcal{R}_a$ capable of performing a, such that $\mathcal{R}_a = \{r \in \mathcal{R} : \exists \sigma \in \mathcal{L}, \exists e \in \sigma$, where $activity(e) = a$ and $resource(e) = r\}$.

4 A Counterfactuals-based Framework for Generating Recommendations

In this section, we aim to propose a framework that leverages counterfactual generations to recommend the next activities and resources.

Figure 2 depicts our proposed framework. The framework is structured into two phases: Offline and Online. The starting point of the offline phase is a completed event log $\mathcal{L}$. From the event log, we build a **transition system** $\mathcal{TS}_{\mathcal{L}}$ (cf. Definition 4) based on the sequence of activities in $\mathcal{L}$. This transition system captures the sequential relationships between activities, representing how tasks are performed one after another. In parallel, the event log is input into a **Trace-to-instance Encoding function** ρ (cf. Section 3). These encoded traces are then used as input to a Predictive Oracle function to estimate the relevant process-related outcomes for a given running instance. In this work, we focus specifically on total execution time prediction, where the corresponding **Total Time Oracle function** is defined in Section 3.

In the online phase, for every running trace σ', let (a_{next}, r_{next}) be the tuple of the next activity and resource that will be followed if no recommendation is given.

[1] For simplicity, events are referred to as activity names, and the event's attributes are abstracted out.

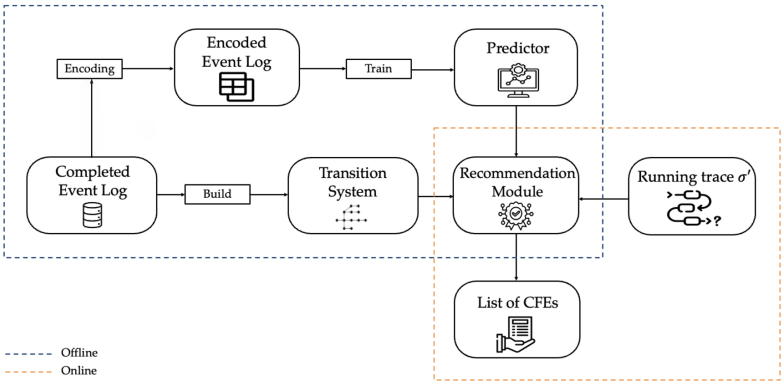


Fig. 2: Schema of our Counterfactual-based Recommender Systems. The schema distinguishes the part executed off-line, namely before activating the system, and that on-line, when recommendations are provided to running cases.

We aim to recommend a suitable activity-resource pair that minimizes the total execution time. The proposed framework takes as inputs a Total Time Oracle function Φ, a transition system $\mathcal{TS}_{\mathcal{L}}$, and a running trace σ'. We first generate a set of k **counterfactual examples** (hereafter referred to as CFEs), $C = \{c_1, c_2, \ldots, c_h\}$ (with $h \leq k$), such that $c_i = (a_i, r_i)$ and each example predicts a different outcome from σ' (i.e., $\Phi(\sigma', a_{next}, r_{next}) > \Phi(\sigma', a_i, r_i)\ \forall i \in \{1, \ldots, h\}$).[2]

Assuming that the transition system $\mathcal{TS}_{\mathcal{L}}$ and the Total Time Oracle function Φ are both constructed from an event log $\mathcal{L}$. First, we build the set of **possible next activities** $\mathcal{A}_{\sigma'}$ that are allowed to occur after observing the events in σ', assuming those coincide with what is observed in $\mathcal{L}$ and thus modeled by $\mathcal{TS}_{\mathcal{L}} : \mathcal{A}_{\sigma'} = \{a : \exists(l^{state}(\sigma'), a, l^{state}(\sigma' \oplus \langle e \rangle))\}$.

The **Counterfactuals Recommendation Module** aims to generate a set C of CFEs, by modifying independently the next activity and resource in the running trace σ'. Each CFE $c_i \in C$ represents a hypothetical scenario in which the next activity-resource pair is replaced with (a_i, r_i), where $a_i \in \mathcal{A}_{\sigma'}$ and $r_i \in \mathcal{R}_{\mathcal{A}_{\sigma'}}$. The expected total execution time for each c_i is predicted through the Total Time Oracle function Φ as defined in Section 3, evaluating $\Phi(\sigma', a_i, r_i)$. However, there is a possibility that the recommended resource r_i is not allowed to perform the activity a_i (i.e., $r_i \notin \mathcal{R}_{a_i}$). In these cases, the recommended action (a_i, r_i) is discarded. The final recommendation is the activity-resource pair (a_{rec}, r_{rec}) that minimizes the expected total execution time (i.e., $(a_{rec}, r_{rec}) = \underset{(a_i, r_i) \in C}{\operatorname{argmin}} \Phi(\sigma', a_i, r_i)$). This ensures that the proposed next activity and resource are both efficient and feasible from the organization's perspective.

Our framework is equipped with a parameter, hereafter referred to as **Reduction Threshold**, that indicates the minimum reduction of process-instance execution times that the recommendation can provide when the activity-resource pair is perturbed. For example, if we set this parameter to 5%, only recommendations that lead to a predicted

[2] In contrast to classification-based counterfactuals, where a change in class label is sufficient, our setting requires a numerically improved (i.e., lower) outcome prediction.

Query instance (original outcome : 2634175.0)

	CLOSURE_TYPE	CLOSURE_REASON	ACTIVITY	RESOURCE	ROLE	...	NEXT_ACTIVITY	NEXT_RESOURCE	TOTAL_TIME
0	Bank Recess	1 - Client lost	Service closure Request with BO responsibility	BOC	BACK-OFFICE	...	Pending Request for Reservation Closure	BOC	2634175.0

Diverse Counterfactual set (new outcome: [0, 2370757])

	CLOSURE_TYPE	CLOSURE_REASON	ACTIVITY	RESOURCE	ROLE	...	NEXT_ACTIVITY	NEXT_RESOURCE	TOTAL_TIME
0	-	-	-	-	-	...	Pending Request for Reservation Closure	574	1992700.0
1	-	-	-	-	-	...	Network Adjustment Requested	0	2154634.0
2	-	-	-	-	-	...	Pending Request for Reservation Closure	SB23	2181635.0

Fig. 3: Example output of the recommendation module with 10% of reduction threshold. The chosen counterfactual has been highlighted in the red box.

reduction in total execution time of *at least* 5% are allowed. It is worth noting that the larger reduction thresholds would generally produce fewer counterfactuals, possibly even none. While the proposed approach supports automated decision making, in scenarios involving a human decision-maker, the system could present the top-k ranked counterfactuals, allowing the user to choose the most appropriate one.

Figure 3 visualizes the output of the recommendation module for a query instance in which a_{next} = "Pending Request for Reservation Closure" and r_{next} = "BOC". The trace is initially predicted to complete in 30 days (2,634,175 seconds); the module aims to reduce execution time by at least 10%, bringing it to a maximum of 27 days (2,370,757 seconds). Therefore, the algorithm modifies the **next_activity** and **next_resource** fields in the original query to explore alternative combinations, generating viable counterfactuals, specifically c_1 = ("Pending Request for Reservation Closure", "574"), c_2 = ("Network Adjustment Requested", "0") and c_3 = ("Pending Request for Reservation Closure", "SB23"). As depicted in the figure, all the counterfactuals satisfy the target range, but c_1 is chosen as it offers the lowest expected execution time.

5 Implementation and Evaluation

This section reports on the results of the proposed counterfactual-based framework for generating recommendations introduced in Section 4. The assessment is conducted using four real-life processes that are outlined in Section 5.1. Section 5.2 discusses the selection of the predictive model for implementing the Total Time Oracle function and reports on the evaluation of its predictive quality across multiple case studies. Section 5.3 provides an overview of the experimental settings and configurations employed for evaluation, while Section 5.4 details the evaluation metrics used. Finally, Section 5.5 reports the results to assess the effectiveness of the recommendations.

Table 1: Descriptive Statistics of Event Logs

Event Log	# cases	# activities	# resources	mean case duration
Bank Account Closure	29194	15	654	16.4 days
BPIC13	7554	13	649	12.1 days
BPIC17AO - Before	12207	18	103	20.2 days
BPIC17AO - After	14164	18	119	21.9 days
Consulta	954	18	561	14.9 days

5.1 Case Studies

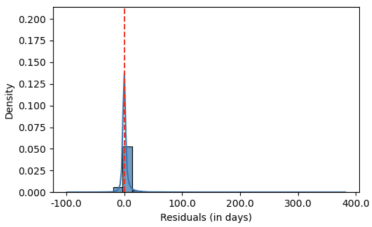
The validity of our framework was assessed using four real-life processes, which are described below. The **Bank Account Closure (BAC)** is a process executed at an banking institution.[3] The process deals with the closure of customer's accounts, which may be requested by the customer or the bank for several reasons. **BPIC13** is a dataset provided by Volvo IT Belgium that contains events from the incident management system.[4] **BPIC17AO** refers to a subprocess that includes application-relevant (A) and offer-relevant activities (O) in the 2017 BPI Challenge. This dataset is derived from a loan application process from a Dutch financial institution.[5] **Consulta** is the Academic Credentials Recognition (ACR) process of a Colombian University that was obtained from its BPM system (Bizagi) for the fourth case study.[6] A concept drift was found in the event log BPIC17AO, in which an increase in the workload of resources at week 22 led to a decrease in the service times at week 28 (cf. [2]). Since our framework assumes no concept drift appears in event logs, we partitioned the event log BPIC17AO into two sub-logs: one contains traces before week 22, namely **BPIC17AO - Before**, and the other contains those after week 28, namely **BPIC17AO - After**. This allows us to experiment across five distinct logs. Table 1 shows the descriptive statistics for each case study, where the mean case duration represents the average total execution time of process instances in that event log.

Under the assumption of no concept drift, each event log is temporally split into a training set $\mathcal{L}^{train}$ (first 75% of the traces) and a test set $\mathcal{L}^{test}$. This approach is useful since predictions rely on past information to forecast future outcomes. Then, we create a test log $\mathcal{L}^{run}$ of running traces to evaluate the effectiveness of the recommendation module. These running traces are derived from $\mathcal{L}^{test}$ by truncating traces at a random point to simulate the scenario of ongoing, incomplete executions.

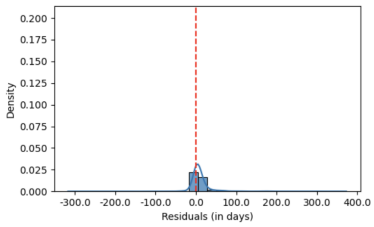
5.2 Prediction Model and Quality

The framework is independent of the specific algorithm to instantiate the predictor of the total execution time of process instances. However, to demonstrate its effectiveness, we instantiated it using CatBoost [11], because of its excellent trade-off between

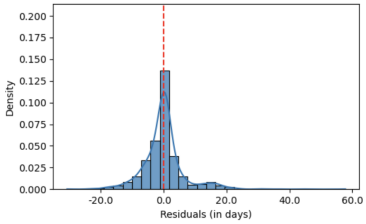
[3] The BAC dataset is confidential and cannot be publicly shared
[4] https://doi.org/10.4121/uuid:500573e6-accc-4b0c-9576-aa5468b10cee
[5] https://doi.org/10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b
[6] https://zenodo.org/records/11067569



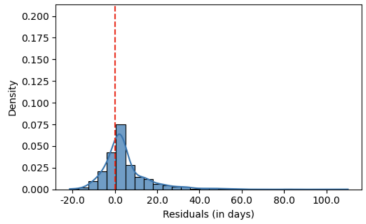
(a) Bank Account Closure



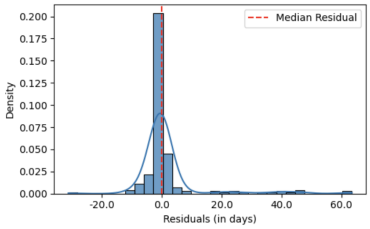
(b) BPIC13



(c) BPIC17AO (Before)



(d) BPIC17AO (After)



(e) Consulta

Case Study	RMAE
BAC	0.215
BPIC13	0.555
BPIC17AO (Before)	0.145
BPIC17AO (After)	0.265
Consulta	0.753

(f) Relative Mean Absolute Error

Fig. 4: Report on prediction quality across different case studies. Subfigures (a) - (e) display the residual distributions (in days) for five case studies: (a) Bank Account Closure (BAC), (b) BPIC13, (c) BPIC17AO (Before), (d) BPIC17AO (After), and (e) Consulta. The dashed red line represents the median residual. Subfigure (f) shows the Relative Mean Absolute Error (RMAE) for each case study.

predictive quality and training time [13] and the presence of an open-source implementation. Additionally, we employed a cross-validation approach to optimize the model's hyperparameters, ensuring robust generalization across different process instances. We randomly picked the 20% of traces in $\mathcal{L}^{train}$ and used them as a validation set $\mathcal{L}^{val}$ to apply a cross-validation approach to optimize the following training hyperparameters of Catboost: *learning_rate, depth, iterations*. The framework has been fully implemented in Python, and the implementation is available on GitHub.[7]

Figures 4a–4e visualize the residual distributions, namely the distribution of the difference between the predicted and actual total execution times for each process instance across the case studies. The nearly symmetric shape of the residuals and the

[7] https://github.com/ngocdiemle296/CounterfactualsPrescriptiveAnalytics

median close to zero highlight the validity of the predictor, indicating that the model is not systematically over- or under-predicting. A model with systematic errors is more predictable, thereby making counterfactual reasoning more effective. The strong performance of the predictors is supported by the relative mean absolute error (RMAE) results as shown in Table 4f, this value is computed by dividing the mean absolute error between the predicted and actual values by the mean of the actual values, providing a normalized measure of prediction error across different case studies. A lower RMAE value indicates higher prediction quality.

5.3 Evaluation Methodology

The whole framework is independent of the specific technique to generate counterfactuals. However, we need to leverage a specific framework, and we opted for DiCE (Diverse Counterfactuals Explanations) [22], which has already been used in predictive process monitoring but only for explanation generation purposes (cf. [7, 15]). In our setup, we used DiCE's random sampling method to generate counterfactuals. The framework allows users to specify the maximum number of generated counterfactuals. In our experiments, we set this limit to 100 to balance between diversity and computational efficiency. However, depending on process constraints and resource availability, fewer counterfactuals may be generated in practice. Unlike many counterfactual methods that focus on producing a single optimal solution, DiCE can generate a diverse set of counterfactuals. This diversity is essential in recommendation contexts, offering a variety of actionable next activity-resource pairs, allowing flexibility in operational decisions. By using the data about historical events and features, DiCE can generate a series of *what-if* scenarios by altering features related to the next activity and resource while keeping others constant.

The reduction threshold parameter of our framework directly maps to a corresponding parameter of DiCE. In our experiment, we evaluate thresholds of 5%, 10%, 20%, and 50%. A higher threshold results in greater potential improvements when a counterfactual is found. However, increasing the threshold also reduces the likelihood of generating recommendations, as more pairs of activity-resource may fail to meet the stricter criteria. By varying these thresholds, we assess the framework's ability to generate diverse recommendations under different levels of execution time reduction.

To evaluate the effectiveness of the proposed method, we compare it against a baseline, referred hereafter to as *Random Resource Assignment (RRA)*. In this method, given a running trace σ', we first extract the list of possible next activities using the transition system. For each activity, we use the Total Time Oracle function to estimate the execution duration and select the activity that minimizes the predicted total time as the next activity. Once the next activity is determined, a resource is randomly chosen from the list of feasible resources to perform that activity.

To assess the effectiveness of our recommendation module, we compare the total execution time of the process instances with and without following recommendations. The best option would be to have an A/B testing with the system running, but that is practically unfeasible. Therefore, building upon prior work in business process simulation [28], this study employs simulation to generate process instances. We developed simulation models for five case studies using established techniques to discover such models [26]. In particular, for each running trace $\sigma' \in \mathcal{L}^{run}$, the simulator is set to the

process state (in fact, a Petri-net marking) that would be reached after performing the activities in $\sigma' \oplus \langle(a, t, r, \emptyset)\rangle$ where the recommendation is to have resource r perform activity a at simulation time t.[8] From the reached state, the process instance was simulated to reach the final state for 10 times, thus mitigating the potential consequences of simulation stochasticity.

Note that the sequence $\sigma_r = \sigma' \oplus \langle(a, t, r, \emptyset)\rangle$ is sometimes not possible in the simulation, namely when the sequence of activities in σ_r is not allowed according the simulation model. If the issue is related to σ' not being possible according to the simulation model, we compute the optimal alignment of σ' [1] to determine the state (i.e., the marking) that would be reached by the execution closest to σ': this state is thus used. If $(a, t, r, \emptyset)$ is not allowed by the simulator in the reached state, the recommendation is discarded, and executions are simulated without following the recommendation.

5.4 Evaluation Metrics

The effect of recommending an activity-resource pair for the running trace is computed as the average total execution time of that trace obtained through simulation. In contrast, the effect of not following the recommendation is the actual average total execution time observed in the test set without applying any intervention.

Let $\mathcal{L}^{test} = \{\sigma_1, \ldots, \sigma_n\}$ be the test set, where each trace $\sigma_i \in \mathcal{L}^{test}$ is associated with its actual total execution time $\mathcal{T}_{total}(\sigma_i)$ for $i \in \{1, ..., n\}$ (cf. Section 3). Let $\mathcal{L}^{run} = \{\sigma'_1, \ldots \sigma'_n\}$ denote the set of running traces, where $\sigma'_i \in prefix(\sigma_i)$. Then, let $\mathcal{L}^{sim} = \{\sigma_1^{sim}, \sigma_2^{sim}, \ldots, \sigma_n^{sim}\}$ be the set of simulated traces where $\sigma_i^{sim} = \sigma'_i \oplus \langle(a_{rec}, t_{rec}, r_{rec}, \emptyset)\rangle \oplus \sigma''_i$ is generated by following recommendation $(a_{rec}, t_{rec}, r_{rec}, \emptyset)$ after σ'_i, and then simulating the continuation σ''_i to reach the final state.

The effectiveness of the recommendations is evaluated based on the improvement gained by following the activity-resource pairs generated by our proposed technique compared to the case where no recommendations are applied. This requires to compute average execution time of the traces in a log - say $\mathcal{L}$ - which is $avg_{time}(\mathcal{L}) = \frac{1}{|\mathcal{L}|} \sum_{\sigma \in \mathcal{L}} \mathcal{T}_{total}(\sigma)$, where $|\mathcal{L}|$ denotes the number of traces in $\mathcal{L}$.

The performance improvement is thus computed as follows:

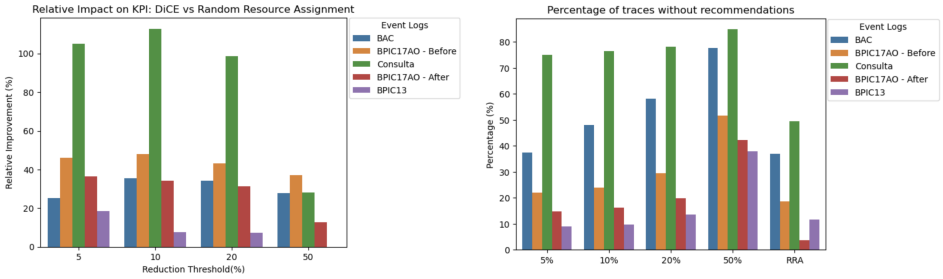
$$\Delta_{perf\_imp}(\mathcal{L}^{test}, \mathcal{L}^{sim}) = \left(1 - \frac{avg_{time}(\mathcal{L}^{sim})}{avg_{time}(\mathcal{L}^{test})}\right)$$

which quantifies the relative gain in total execution time achieved by applying our method over the ground truth performance, which was observed in the event log when no recommendations are provided. We introduce a metric called **relative improvement**, Δ_{rel_imp}, which measures the effectiveness of our method compared to the RRA approach:

$$\Delta_{rel\_imp}(\mathcal{L}^{test}, \mathcal{L}^{sim}_{counter}, \mathcal{L}^{sim}_{RRA}) = 1 - \frac{\Delta_{perf\_imp}(\mathcal{L}^{test}, \mathcal{L}^{sim}_{counter})}{\Delta_{perf\_imp}(\mathcal{L}^{test}, \mathcal{L}^{sim}_{RRA})}$$

where $\mathcal{L}^{sim}_{counter}$ and $\mathcal{L}^{sim}_{RRA}$ are the event logs obtained via simulation when the recommendations are produced by our method based on counterfactual generations and by the RRA method, respectively.

[8] Symbol $\emptyset$ indicates a function with an empty domain.



(a) (b)

Fig. 5: (a) Relative improvement related to the average total execution time when applying DiCE compared to the Random Resource Assignment method. On the x-axis, there are the reduction thresholds, while on the y-axis, the relative improvement is computed on the whole $\mathcal{L}^{run}$. (b) Percentage of traces without recommendations when applying DiCE compared to the Random Resource Assignment method. On the x-axis, there are the percentages of reduction thresholds (5%, 10%, 20%, and 50%) when applying DiCE and the RRA column indicating the Random Resource Assignment method.

5.5 Results Analysis

This section reports on the main results of our counterfactuals recommendation module.

Figure 5a presents a comparative visualization of the relative improvement achieved by our framework over the RRA method. Overall, our framework outperforms the RRA approach, with particularly substantial gains observed in the Consulta process, where improvements exceed 100% for the three lowest reduction thresholds. This confirms how a poor choice of a resource to perform a recommendation activity may significantly affect performance and how our framework efficiently generates recommendations in the form of activities and corresponding allocated resources. A noticeable trend is observed where the relative improvement for all methods declines at the 50% reduction threshold. Moreover, when looking at the BPIC13 process, we observe that the proposed method is not able to provide any relative improvement at the 50% total execution time. The underlying reasons for this behavior will be discussed in the following paragraph.

Another aspect we want to take into account is the percentage of traces without recommendations. This is important because the effectiveness of our framework is not solely determined by the performance improvement but also by how many traces can actually be optimized. This may be attributed to two different causes: (i) the model's threshold is set too high, resulting in no recommendations that meet this criterion, or, conversely, (ii) the ongoing trace already represents the optimal execution for the model, making further improvements practically unfeasible. From Figure 5b, we can notice that if the percentage of the reduction threshold increases, there will be a decrease in the percentage of traces that receive recommendations. This highlights that *if the threshold is pushed too high, several process instances remain without potential recommendations*, which implies that the overall improvement for all running instances is more limited. Since the framework generates recommendations based on patterns and relationships identified in the event log data, setting a high percentage of reduction thresholds may

cause the algorithm to focus only on the most extreme cases - those that can achieve significant improvements. As a result, the algorithm becomes more selective, filtering out traces that do not meet the strict total time criteria. This leads to a decrease in the number of eligible traces for recommendations. We are aware that choosing a higher percentage for reduction may seem very optimistic. However, it requires balancing the trade-off between achieving significant execution time improvements and finding a suitable number of recommendations for a substantial number of running cases.

As shown in Figure 5b, the RRA approach can consistently generate a higher number of recommendations. This is because RRA does not rely on complex optimization techniques. Instead, it selects the next activity based on the smallest predicted total time and assigns a resource randomly based on the chosen best activity. This random allocation often results in suboptimal resource utilization.

In conclusion, our framework can effectively recommend activity-resource pairs that significantly reduce the total execution time, even under strict constraints, resulting in substantial improvements across all five event logs and reduction thresholds. Meanwhile, the random resource assignment prevents the RRA method from achieving equally good results, especially when resource choice critically impacts outcomes. The experiments show that setting overly high reduction thresholds may limit applicability, as many traces cannot receive recommendations. This highlights a key strength of the framework: the explicit tuning of the reduction threshold enables process experts to balance applicability - the proportion of traces that can receive recommendations - and utility - the magnitude of the improvements achieved through these recommendations.

6 Conclusion

This paper has put forward a framework for prescriptive process analytics that utilizes counterfactuals to generate real-time recommendations for the next resource and activity. In particular, DiCE was employed as a framework to generate these counterfactuals. In prescriptive process analytics, recommendations need to be given, usually in the form of suggesting a certain activity to be assigned to a resource for those process instances that are predicted to likely complete with unsatisfactory outcomes. In this paper, we focus on total case duration as the outcome, which is aimed to be minimized.

The idea of using counterfactuals for generating recommendations in prescriptive process analytics has stemmed from the idea that processes usually consist of dozens of potential activities and even hundreds of potential resources to whom these activities can be assigned. While brute-force approaches are not applicable in practice with these numbers, several existing approaches use a divide-and-conquer approach in which, first, the activity is chosen, and then it is assigned to a suitable resource. The leverage of counterfactual generation has the advantage of relying on more innovative approaches to converge to highly effective recommendations quickly.

Experiments were conducted on several real-life processes. These showed that our technique, based on counterfactual generation, outperforms a natural baseline where the best activities are first chosen and then allocated to resources. Experiments also highlight the framework's ability to be fine-tuned, providing a proper balance between applicability, referring to the proportion of traces that receive recommendations, and utility, which reflects the extent of the improvements achieved.

We acknowledge that this is the first work in the literature to provide recommendations based on counterfactual generation, and several avenues for future work exist. Currently, we aim to reduce the execution time since, although simplistic, it is a general indicator desired across various domains. However, extending it to other KPIs is conceptually simple. Furthermore, since the proposed technique does not depend on the method used for generating counterfactuals, we also plan to experiment with alternative counterfactual-generation frameworks, such as large language models, for explaining/producing recommendations about activities and resources.

Acknowledgments. The Ph.D. scholarships of Ngoc-Diem Le and Francesco Vinci are financially supported by MUR (PNRR) and the University of Padua. Authors also acknowledge the support of the project "Future AI Research (FAIR) - Spoke 2 Integrative AI - Symbolic conditioning of Graph Generative Models (SymboliG)" funded by the European Union under the National Recovery and Resilience Plan (NRRP), Mission 4 Component 2 Investment 1.3 - Call for tender No. 341 of March 15, 2022 of Italian Ministry of University and Research – NextGenerationEU, Code PE0000013, Concession Decree No. 1555 of October 11, 2022 CUP C63C22000770006. The authors gratefully acknowledge Dr. Massimiliano Ronzani and Mr. Andrei Buliga, from Fondazione Bruno Kessler - Trento, for their valuable feedback provided during the final stages of manuscript preparation.

References

1. van der Aalst, W.M.P.: Process Mining: Data Science in Action. Springer (2016)
2. Adams, J.N., van Zelst, S.J., Quack, L., Hausmann, K., van der Aalst, W.M., Rose, T.: A framework for explainable concept drift detection in process mining. In: Proceedings of the 19th International Conference on Business Process Management (BPM 2021). pp. 400–416 (2021)
3. Arias, M., Rojas, E., Munoz-Gama, J., Sepúlveda, M.: A framework for recommending resource allocation based on process mining. In: Proceedings of the 13th International Conference on Business Process Management (BPM 2015) (2015)
4. Bhattacharya, A., Ooge, J., Stiglic, G., Verbert, K.: Directive explanations for monitoring the risk of diabetes onset: Introducing directive data-centric explanations and combinations to support what-if explorations. In: Proceedings of the 28th International Conference on Intelligent User Interfaces (IUI '23). pp. 204–219. IUI '23 (2023)
5. Bozorgi, Z.D., Teinemaa, I., Dumas, M., Rosa, M.L., Polyvyanyy, A.: Prescriptive process monitoring for cost-aware cycle time reduction. In: Proceedings of the 3rd International Conference on Process Mining (ICPM 2021). pp. 96–103 (2021)
6. Branchi, S., Di Francescomarino, C., Ghidini, C., Massimo, D., Ricci, F., Ronzani, M.: Learning to act: a reinforcement learning approach to recommend the best next activities. In: Proceedings of the 20th International Conference on Business Process Management (BPM 2022). pp. 137–154. Springer (2022)
7. Buliga, A., Di Francescomarino, C., Ghidini, C., Maggi, F.M.: Counterfactuals and ways to build them: Evaluating approaches in predictive process monitoring. In: Proceedings of the 35th International Conference on Advanced Information Systems Engineering (CAiSE 2023). p. 558–574. Springer-Verlag, Berlin, Heidelberg (2023)
8. Buliga, A., Di Francescomarino, C., Ghidini, C., Montali, M., Ronzani, M.: Generating counterfactual explanations under temporal constraints. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 39, pp. 15622–15631 (2025)
9. Comuzzi, M.: Ant-colony optimisation for path recommendation in business process execution. Journal of Data Semantics **8**(2), 113–128 (2019)
10. Donadello, I., Di Francescomarino, C., Maggi, F.M., Ricci, F., Shikhizada, A.: Outcome-oriented prescriptive process monitoring based on temporal logic patterns. Engineering Applications of Artificial Intelligence **126**, 106899 (2023)

11. Dorogush, A.V., Ershov, V., Gulin, A.: Catboost: gradient boosting with categorical features support. In: Proceedings of the Workshop on ML Systems at NIPS 2017 (2017)
12. Fahrenkrog-Petersen, S., Tax, N., Teinemaa, I., Dumas, M., de Leoni, M., Maggi, F., Weidlich, M.: Fire now, fire later: alarm-based systems for prescriptive process monitoring. Knowledge and Information Systems **64** (02 2022)
13. Galanti, R., de Leoni, M., Monaro, M., Navarin, N., Marazzi, A., Di Stasi, B., Maldera, S.: An explainable decision support system for predictive process analytics. Engineering Applications of Artificial Intelligence **120**, 105904 (2023)
14. Guidotti, R.: Counterfactual explanations and how to find them: literature review and benchmarking. Data Mining and Knowledge Discovery **38**(5), 2770–2824 (2024)
15. Hsieh, C., Moreira, C., Ouyang, C.: DiCE4EL: Interpreting process predictions using a milestone-aware counterfactual approach. In: Proceedings of the 3rd International Conference on Process Mining (ICPM 2021). pp. 88–95. IEEE (2021)
16. Huang, T.H., Metzger, A., Pohl, K.: Counterfactual explanations for predictive business process monitoring. In: Proceedings of EMCIS 2021. Springer (2022)
17. Hundogan, O., Lu, X., Du, Y., Reijers, H.A.: Created: Generating viable counterfactual sequences for predictive process analytics. In: Proceedings of the 35th International Conference on Advanced Information Systems Engineering (CAiSE 2023) (2023)
18. Käppel, M., Ackermann, L., Jablonski, S., Härtl, S.: Attention please: What transformer models really learn for process prediction. In: Proceedings of the 22nd International Conference on Business Process Management (BPM 2024). pp. 203–220. Cham (2024)
19. Kubrak, K., Milani, F., Nolte, A., Dumas, M.: Prescriptive process monitoring: Quo vadis? PeerJ Comput. Sci. **8**, e1097 (2022)
20. Li, J., Ren, Y., Deng, K.: Fairgan: Gans-based fairness-aware learning for recommendations with implicit feedback. In: Proceedings of the the Web Conference 2022. p. 297–307 (2022)
21. Mehdiyev, N., Majlatow, M., Fettke, P.: Counterfactual explanations in the big picture: An approach for process prediction-driven job-shop scheduling optimization. Cognitive Computation **16**(5), 2674–2700 (May 2024)
22. Mothilal, R.K., Sharma, A., Tan, C.: Explaining machine learning classifiers through diverse counterfactual explanations. In: Proceedings of the 3rd ACM Conference on Fairness, Accountability, and Transparency (ACM FAT 2020). pp. 607–617 (2020)
23. Navarin, N., Vincenzi, B., Polato, M., Sperduti, A.: LSTM networks for data-aware remaining time prediction of business process instances. In: Proceedings of the IEEE Symposium Series on Computational Intelligence (SSCI 2017) (2017)
24. Padella, A., de Leoni, M.: Resource allocation in recommender systems for global kpi improvement. In: Proceedings of the 21st International Conference on Business Process Management (BPM 2023) (2023)
25. Padella, A., de Leoni, M., Dogan, O., Galanti, R.: Explainable process prescriptive analytics. In: Proceedings of the 4th International Conference on Process Mining (ICPM 2022). IEEE (2022)
26. Padella, A., Mannhardt, F., Vinci, F., de Leoni, M., Vanderfeesten, I.: Experience-based resource allocation for remaining time optimization. In: Proceedings of the 22nd International Conference on Business Process Management (BPM 2024) (2024)
27. Shoush, M., Dumas, M.: White box specification of intervention policies for prescriptive process monitoring. Data Knowl. Eng. **155**, 102379 (2025)
28. Shoush, M., Dumas, M.: Prescriptive process monitoring under resource constraints: A reinforcement learning approach. KI - Künstliche Intelligenz (2024)
29. Verma, S., Dickerson, J., Hines, K.: Counterfactual explanations for machine learning: A review. arXiv preprint arXiv:2010.10596 **2**(1), 1 (2020)
30. Wachter, S., Mittelstadt, B., Russell, C.: Counterfactual explanations without opening the black box: Automated decisions and the gdpr. Harvard Journal of Law & Technology **31**, 842–887 (2017)