

Balancing KPIs through Multi-Objective Prescriptive Process Analytics

Ngoc-Diem Le¹, Andrei Buliga², Massimiliano Ronzani², Alessandro Padella¹, and
Massimiliano de Leoni¹

¹ University of Padua, Padua, Italy

² Fondazione Bruno Kessler, Trento, Italy

ngocdiem.le@phd.unipd.it, abuliga@fbk.eu, mronzani@fbk.eu,
alessandro.padella@unipd.it, deleoni@math.unipd.it

Abstract. Prescriptive Process Analytics aims to support ongoing process executions by guiding them toward desirable outcomes, which are usually measured through Key Performance Indicators (KPIs). While the literature on Prescriptive Process Analytics has focused on the exploration of different recommendation techniques and tasks, the current efforts have so far largely focused on optimizing a single KPI. However, real-world processes often involve multiple, interrelated, and potentially conflicting KPIs, requiring an explicit treatment of their trade-offs. This paper presents a framework that generates recommendations designed to optimize multiple KPIs simultaneously, and that allows users to explore different levels of trade-off among them. Our proposed framework jointly recommends the next activity and its executing resource, which we call the next step recommendation, aiming to balance multiple KPIs. Different strategies are proposed to generate these recommendations, including one that utilizes a genetic algorithm to efficiently explore the search space of viable options. This is particularly effective in domains with a large number of resources, which lead to a significantly extensive space of possible recommendations. Experimental results demonstrate that our multi-KPI approach achieves favorable trade-offs among partially conflicting goals, while minimizing the degradation of each individual KPI compared to single-objective optimization.

Keywords: Process Mining · Multi-objective Optimization · Prescriptive Process Analytics

1 Introduction

Prescriptive Process Analytics (PrPA) [23] aims to provide decision-makers with actionable insights to prevent “risky” process executions from resulting in undesired outcomes, or to optimize specific aspects of ongoing process execution, such as minimizing execution time or maximizing profit. The field of PrPA has gained increasing interest in recent years, employing a variety of techniques and targeting different prescription types (cf. Section 2).

Existing approaches have predominantly addressed situations where recommendations aim to improve one specific Key Performance Indicator (KPI), such as avoiding undesired outcomes (e.g., an application rejection) or reducing time or cost. How-

ever, in real-world scenarios, multiple KPIs are typically relevant for process optimization, and they are often interdependent and sometimes conflicting. For example, hospital managers may seek policies that shorten patients' hospital stays to reduce costs while still maintaining, or even improving, post-discharge outcomes, such as lower re-hospitalization rates, even though these two goals may be in conflict.

It thus follows that identifying an appropriate trade-off among the different KPIs becomes essential, as the optimal trade-off between multiple KPIs depends on many extrinsic factors, such as the process domain or the given process variability.

Many existing works have overlooked this aspect, and only a few have attempted to address the multi-objective nature of business process optimization problems by defining complex KPIs that combine multiple factors [6]. However, even these approaches typically rely on predefined formulas to merge objectives, without investigating the intrinsic trade-offs between them, or allowing users a method in which these trade-offs can be explored within the process.

In this work, we take a first step toward addressing this gap by proposing a framework for prescriptive process analytics in a multi-KPI optimization setting, explicitly accounting for the trade-offs among competing KPIs. In this way, decision-makers can explore different trade-offs between the KPIs and determine which prescription best suits their requirements. We frame the prescription as a combined recommendation of both activity and resource, which we call *next step recommendation*, which has already been shown to be effective in improving time-oriented objectives [24]. Given the potentially very large search space for this type of recommendation, especially when hundreds of resources are present in the event log, the problem of computational scalability is also addressed within this paper.

The contributions of our work are as follows: (i) a new multi-objective framework for process next step recommendation, enabling simultaneous consideration of multiple, potentially conflicting KPIs; (ii) an efficient technique that employs a genetic algorithm to efficiently navigate the search space of activity-resource pairs, while preserving the effectiveness of the recommendations; (iii) extensive experimental evidence demonstrating that multi-objective optimization can effectively balance partially conflicting objectives.

The remainder of this paper is structured as follows: Section 2 discusses the related work in prescriptive process analytics. Section 3 introduces the background concepts, and Section 4 proposes a framework that leverages the prediction-based approach for generating recommendations. The case studies and the evaluation setup are described in Section 5, while Section 6 reports the results and their limitations. Finally, Section 7 concludes the paper by summarizing our contribution and discussing future directions.

2 Related Work

In the field of process mining, PrPA has gained significant attention over the past decade [23]. Earlier approaches focused on recommending the next activity to execute by leveraging available data payloads and basic forms of counterfactual reasoning [20], aiming to improve overall business process performance. Subsequently, in parallel with the rise of prediction-based methods [26], the focus of PrPA expanded toward anticipat-

ing which ongoing process instances might lead to unfavorable KPIs [38]. These developments enabled the prescription of corrective actions and investigations into optimal intervention timing [18], also providing general guidance for process continuation [10].

Growing adoption of machine- and deep-learning techniques has enabled the enhancement of prescriptive capabilities [29, 41]. Research in this phase explored the integration of causal inference [4, 5], reinforcement learning [6, 28, 36], and white-box approaches [33, 37]. Additional progress was achieved by exploiting correlations between temporal relations [13], resource behavior [35], and enhanced computational efficiency through gradient boosting [33, 37].

Recent advancements have focused on approaches that optimize a given KPI through recommending the optimal resource allocation [27], or jointly recommending both the activity to perform and the resource responsible for its execution [32], also accounting for resource experience curve [34]. Moreover, recent work has begun to integrate Large Language Models to enhance the interpretability of recommendations [22].

Prior decision-support framework has leveraged Business Process Simulation (BPS) and visual analytics to facilitate the manual exploration of complex, context-aware “what-if” scenarios [3]. Recent approach combines generative AI and BPS to synthesize these scenarios while preserving all process constraints [8]. Since the evaluation of these scenarios relies on BPS, their effectiveness depends on the accuracy of the underlying models. Consequently, recent study focuses on explicitly modeling extraneous activity delays [9], and using online discovery methods to ensure simulations remain robust in evolving process environments [40].

Despite these advancements, current PrPA frameworks primarily aim to optimize a single predefined KPI, often at the cost of other relevant dimensions that are neglected in the recommendation process. Even previous works, such as [6], which consider multiple process aspects, including time and control flow, combine them into a rigid KPI definition with a specific interpretation (e.g., profit), without allowing the exploration of different trade-offs between objectives. To address this limitation, the present work introduces a framework that optimizes multiple KPIs simultaneously, weighting their relative importance according to user-defined preferences.

3 Preliminaries and Problem Definition

In this section, we introduce the basic concepts required to understand the remainder of the paper.

3.1 Event, Traces, Event Log

An event log $\mathcal{L}$ is defined as a collection of execution sequences of a business process, known as *traces*. Each trace σ consists of a series of chronologically ordered events, $\sigma = \langle e_1, e_2, \dots, e_n \rangle$. Given a complete trace σ , we define its k -prefix, with $k < |\sigma|$,³ as the sequence of the first k events $\sigma_{(k)} = \langle e_1, \dots, e_k \rangle$.

Each event e corresponds to the execution of an activity a and is uniquely associated with a trace through a *Case ID*, indicated as c . Therefore, each event is defined as $e =$

³ Given a trace σ , $|\sigma|$ indicates the number of events in σ .

$(c, a, \mathbf{d})$, where $\mathbf{d}$ is a vector representing additional event attributes. These attributes, also referred to as the event's data payload, describe characteristics of the case or of the event itself. Common examples include timestamps marking the start and completion of the activity execution, as well as the resource r performing it.

In this work, we focus on prescriptions involving the activity to be executed and its associated resource. To this aim, we denote by $\mathcal{A}$ and $\mathcal{R}$ the sets of all activities and resources available in the process. We also define the set of all possible events as $\mathcal{E}$ and the space of event sequences as $\mathcal{E}^*$. Finally, we define the projection functions $act : \mathcal{E} \rightarrow \mathcal{A}$ and $res : \mathcal{E} \rightarrow \mathcal{R}$, which map each event e to its corresponding activity a and resource r , respectively.

3.2 Problem Definition

In this work, we address the problem of multi-KPI optimization in PrPA. These KPIs represent relevant aspects of process performance and are determined by the specific application domain.

A KPI can be modelled as a function $F : \mathcal{E}^* \times \mathbb{N} \rightarrow \mathbb{R}$ that, given a trace $\sigma \in \mathcal{E}^*$ and an integer $1 \leq i \leq |\sigma|$, $F(\sigma, i)$ returns the KPI value of σ after the occurrence of the first i events. To account for multiple KPIs $F_1, \dots, F_n$, we adopt the *weighted global criterion* method [25], in which all KPIs are combined linearly to form a single objective function, called the *utility function*. Let $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_n) \in [0, 1]^n$ be a vector of weights such that $\sum_{j=1}^n \lambda_j = 1$. These weights allow decision-makers to adjust the trade-off among different KPIs, reflecting both domain priorities and temporary or situational preferences. The utility function is then defined as:

$$U_{\text{KPI}}(\sigma, \boldsymbol{\lambda}, i) = \sum_{j=1}^n \lambda_j F_j(\sigma, i) \quad (1)$$

The prescriptive problem can be formulated as follows. Given a running process execution $\sigma' = \langle e_1, \dots, e_k \rangle$, we aim to find a prescription corresponding to e_{k+1} that will eventually lead to a completed trace $\sigma_{\text{pres}} = \langle e_1, \dots, e_k, e_{k+1}, \dots, e_n \rangle$, such that the utility function $U_{\text{KPI}}(\sigma, \boldsymbol{\lambda}, i)$ is maximized.

4 Framework

In this section, we present a framework to address the problem of next-step recommendation for multi-KPI optimization, as defined in Section 3.2. The framework consists of three main modules: (i) a *feasibility prescription module* (Section 4.1); (ii) a *predictive model* (Section 4.2); and (iii) a *recommendation module* (Section 4.3). As shown in Figure 1, the proposed framework comprises two phases: **Offline** and **Online**. The offline phase starts with an event log of completed executions, which is used to build the feasibility prescription module to identify the set of admissible next-step prescriptions. Concurrently, the predictive model is trained on this historical log to estimate the expected value of the utility function (U_{KPI}) for any given prescription. These two components are then combined to build the recommendation module, which is utilized

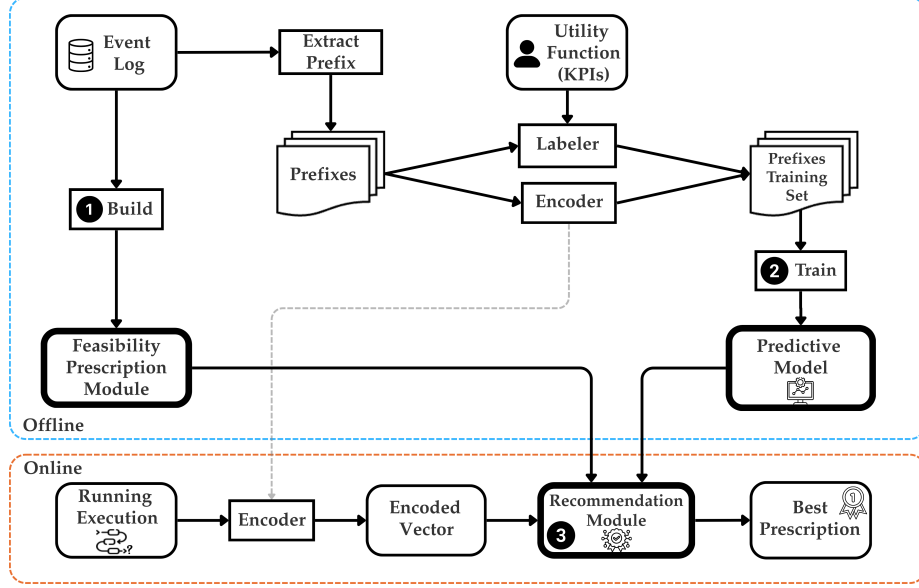


Fig. 1: Overview of the proposed framework. The offline phase comprises: (1) building the Feasibility Prescription Module from historical event log and (2) training a Predictive Model using Prefixes Training Set transformed by the Labeler and Encoder. The online phase performs real-time recommendation of a Running Execution by identifying the Best Prescription through the (3) Recommendation Module.

during the online phase. In this phase, each running execution (or prefix) is provided as input, and transformed into an encoded vector. The recommendation module then searches the space of possible prescriptions to identify the most promising alternatives to recommend to the user as the best prescription. We propose two methods for exploring the prescription space: an exhaustive search that evaluates all possible prescriptions, and a more sophisticated search based on a genetic algorithm.

In the following, we detail all the modules used in our framework.

4.1 Feasibility Prescription Module

The feasibility prescription module, corresponding to Step 1 in the offline phase of Figure 1, is built directly from the historical event log. The primary purpose of the module is to constrain the recommendation space to only those actions that are practically executable. Therefore, this module acts as a validation layer to assess the eligibility of a potential activity and its assigned resource based on the process state. This is formalized in our study as a *transition system* to determine if, given the current prefix $\sigma_{(k)}$, an activity a_{k+1} is valid for execution, and a *resource filter* to confirm that the resource r_{k+1} is eligible to perform that specific activity.

The transition system is mined from a log of completed traces $\mathcal{L}$, from which each prefix $\sigma_{(k)}$ is mapped to an abstract state representation $S(\sigma_{(k)})$ [1]. Figure 2 illustrates

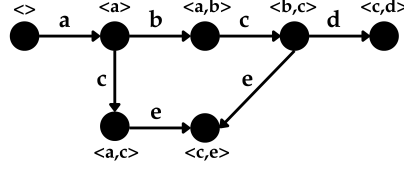


Fig. 2: Transition system for an example log $\mathcal{L}^{\text{ex}}$, with window size $w = 2$.

an example of a transition system. It has been built on an event log $\mathcal{L}^{\text{ex}}$ using a sequence-based state-representation where $\mathcal{L}^{\text{ex}} = \{\langle a, b, c, d \rangle, \langle a, c, e \rangle, \langle a, b, c, e \rangle\}$, $a, b, c, d, e \in \mathcal{A}$ form the alphabet of activity labels⁴.

The set of feasible next activities for a given prefix $\sigma_{(k)}$ is defined as those activities that directly follow the corresponding abstract state at least once in the event log, as

$$\mathcal{A}|_{\sigma_{(k)}} = \{a \in \mathcal{A} \mid \exists \sigma' \in \mathcal{L}, \exists l < |\sigma'| : S(\tilde{\sigma}_{(l)}) = S(\sigma_{(k)}), a = \text{act}(\sigma'_{(l+1)})\}. \quad (2)$$

The abstraction of the prefix is necessary both to prevent the transition system from being overly deterministic and to avoid an explosion in the number of states. Our choice for the state abstraction of the prefix $\sigma_{(k)}$ corresponds to the list of the last w activities seen in the prefix, that is

$$S^w(\sigma_{(k)}) = \begin{cases} \langle a_{k-w+1}, \dots, a_k \rangle & \text{if } w \leq k; \\ \langle a_1, \dots, a_k \rangle & \text{otherwise.} \end{cases} \quad (3)$$

Considering the example event log $\mathcal{L}^{\text{ex}}$ and the prefix $\sigma_{(k)} = \langle b, b, c \rangle$. The state representation (3) with window size $w = 2$ is defined as $S^{w=2}(\sigma_{(k)}) = \langle b, c \rangle$ and the set of feasible next activities for it is $\mathcal{A}|_{\sigma_{(k)}} = \{d, e\}$.

Furthermore, we determine whether a resource r is capable of performing an activity a based on evidence from the event log $\mathcal{L}$; specifically, there must exist an event $e \in \mathcal{L}$ such that $\text{act}(e) = a$ and $\text{res}(e) = r$. Given a prefix $\sigma_{(k)}$, for each activity $a \in \mathcal{A}|_{\sigma_{(k)}}$, we then identify the set of *eligible resources* $\mathcal{R}_a$ that are able to execute a , defined as follows: $\mathcal{R}_a = \{r \in \mathcal{R} \mid \exists \sigma \in \mathcal{L}, \exists e \in \sigma, \text{act}(e) = a, \text{res}(e) = r\}$. Finally, we define the *space of feasible next step prescriptions* for the prefix $\sigma_{(k)}$ as $\mathcal{P}_{\sigma_{(k)}} = \{(a, r) \mid a \in \mathcal{A}|_{\sigma_{(k)}}, r \in \mathcal{R}_a\} \subseteq \mathcal{A} \times \mathcal{R}$.

4.2 Predictive Model for KPIs Estimation

To estimate which feasible next-step prescription is likely to yield the greatest improvement across multiple KPIs, we leverage Predictive Process Monitoring (PPM) [12]. As shown in the offline phase of Figure 1 (Step 2), we train a predictive model ϕ that, given a running trace, or prefix, $\sigma_{(k)}$ and a candidate next-step prescription (a_{k+1}, r_{k+1}) , with $a_{k+1} \in \mathcal{A}|_{\sigma_{(k)}}$ and $r_{k+1} \in \mathcal{R}_{a_{k+1}}$, estimates the value of the utility function U_{KPI} upon completion of the process execution. The model corresponds to a function $\phi : \mathcal{E}^* \times \mathcal{A} \times \mathcal{R} \rightarrow \mathbb{R}$ trained as a regressor on the training log. For each trace $\sigma \in \mathcal{L}$

⁴ For simplicity, events are referred to as activity labels, omitting the event attributes.

and each step $1 \leq k < |\sigma|$, the training objective is $\phi(\sigma_{(k)}, act(e_{k+1}), res(e_{k+1})) \approx U_{KPI}(\sigma, \lambda, k)$, where the weights λ are set by the user, as discussed in Section 3.2.

To leverage PPM techniques for building the predictive model, it is necessary to encode the input information into a structured numeric representation. Our framework is agnostic to the specific choice of encoding; however, in this work we adopt the following approach. For the prefix $\sigma_{(k)}$, we employ *activity frequency encoding*, a commonly used representation in PPM tasks [12, 19, 33, 35], in which the prefix is described by a vector counting the occurrences of each activity up to the current step. Additionally, we encode auxiliary attributes, including the prefix duration, the duration of the current event⁵, and the weekday corresponding to the most recent event timestamp. Finally, the next activity a_{k+1} and next resource r_{k+1} are represented using label encoding.

To ensure consistency between training and model inference, the same encoding logic used to build the Prefixes Training Set is applied to the Running Execution during the online phase.

4.3 Recommendation Module

The final component of the framework is the Recommendation Module (Step 3), which is used in the online phase to explore the space of feasible next-step prescriptions $\mathcal{P}_{\sigma_{(k)}}$ for the prefix $\sigma_{(k)}$, as defined in Section 4.1. In the following, we present two alternative approaches to perform this exploration.

Exhaustive Search. The first approach is to exhaustively explore all pairs of feasible prescriptions $(a, r) \in \mathcal{P}_{\sigma_{(k)}}$. For each pair, the predictive model described in Section 4.2 estimates the future value of the utility function U_{KPI} assuming that prescription is applied, yielding $\phi(\sigma_{(k)}, a, r)$. The candidate prescriptions are then ranked according to this estimated value, and the best one is returned to the user as the recommended next step. In this way, the selected prescription corresponds to the best possible decision according to the predictive model. However, this approach may become computationally inefficient, especially in processes characterized by a large number of activities and resources, corresponding to a large search space $\mathcal{P}_{\sigma_{(k)}}$.

Genetic Method. To improve the computational efficiency of exploring feasible prescriptions, we leverage a genetic algorithm [30], which iteratively explores a subset of the search space $\mathcal{P}_{\sigma_{(k)}}$. Specifically, an *initial population* of fixed size p of feasible prescriptions is randomly extracted, $P_0 \subset \mathcal{P}_{\sigma_{(k)}}$ with $|P_0| = p$. The population evolves through several iterations P_i with $i \in \mathbb{N}$, through a sequence of evolutionary steps. At each iteration i , the algorithm evaluates the quality of every prescription pair $(a, r) \in P_i$ using the fitness function.

The objective of the genetic algorithm is to obtain a final population that minimizes the fitness function, which evaluates two important aspects for each candidate: *validity* and *proximity*. Validity *val* measures whether the predicted KPIs' utility function, as

⁵ The current event duration is computed as the difference between the event's end and start timestamps. If the start timestamp is not available, it is approximated as the difference between the end timestamp of the current event and that of the previous event.

estimated by the predictive model ϕ for the prescription (a, r) applied to $\sigma_{(k)}$, improves by at least a threshold $\tau \in [0, 1]$ compared to the predicted value for $\sigma_{(k)}$ without any prescription. Namely, $val(a, r) = 0$ if $\phi(\sigma_{(k)}, a, r) < (1 - \tau)\phi(\sigma_{(k)}, \cdot, \cdot)$ otherwise $val(a, r) = 1$. Proximity $prox$ measures the difference between a candidate pair (a, r) the most probable activity-resource pair $(\hat{a}, \hat{r})$ observed in the training event log following the input prefix $\sigma_{(k)}$, taking the following values:

$$prox(a, r) = \begin{cases} 0, & \text{if } a = \hat{a} \text{ and } r = \hat{r}, \\ 1, & \text{if } (a = \hat{a} \wedge r \neq \hat{r}) \text{ or } (a \neq \hat{a} \wedge r = \hat{r}), \\ 2, & \text{if } a \neq \hat{a} \text{ and } r \neq \hat{r}. \end{cases}$$

Proximity helps prioritize pairs that are closer to typical patterns while still allowing exploration of alternatives.

Inspired by [7], we define our fitness function as follows:

$$fitness(a, r) = val(a, r) + \alpha prox(a, r), \quad (4)$$

where α is the weighting factor that controls the influence of the proximity score on the overall fitness. The fitness score, as defined in Equation 4, quantifies how promising each candidate is. Based on this value, the algorithm selects the 50% most suitable pairs in the population to serve as parents for the next generation. These parent solutions then undergo a set of genetic operators, namely crossover and mutation, where their information is perturbed to produce the new population. Crossover combines the chromosomes of the parents to generate new offspring, while mutation randomly alters one or more genes with a certain probability, introducing new genetic material and helping to prevent premature convergence. The resulting offspring form the new population. Once the evolutionary process converges or reaches the maximum number of iterations, the evolutionary algorithm returns the best m activity-resource pairs that minimize the fitness function. The recommended prescription is then the one that minimizes the prediction for U_{KPI} obtained from the predictive model ϕ .

5 Evaluation Methodology

This section reports on the methodology used to assess the effectiveness of the framework introduced in Section 4.⁶ The evaluation is conducted using four real-life case studies, outlined in Section 5.1. Then, Section 5.2 describes the experimental settings, including data splitting and the implementation of the Genetic approach. Section 5.3, discusses the selection of the predictive model and evaluate its quality across multiple case studies, while Section 5.4 provides an overview of how the recommendations are evaluated. The evaluation aimed to address the following research questions:

RQ1 What is the impact of using the Genetic method in terms of KPIs improvement and computation time, compared to the Exhaustive method?

⁶ The framework and evaluation are available in the following repository https://github.com/ngocdiemle296/multi_kpi_prescriptive_process_analytics.

RQ2 How does the weights' vector $(\lambda_1, \dots, \lambda_n)$, defined in Equation 1, affect the trade-offs between the individual KPIs, under the proposed strategies?

In business process analysis, the Devil's Quadrangle illustrates that improvements in one performance dimension may negatively affect others [17]. Since remaining time reflects the time dimension, while outcome-based measures represent the quality dimension [21], the two KPIs may naturally move in different directions. Therefore, this motivates us to frame our evaluation to target two potentially conflicting KPIs: the *remaining execution time* and the *case outcome*. The case outcome denotes the final status of a trace, expressed as a boolean value; for example, it may be defined by the occurrence of a key activity or the satisfaction of a condition on an event attribute. The conflicting nature of these two KPIs provides a controlled setting to observe how different λ_i values influence the trade-offs in optimization. For example, recommending the "cancel application" activity in the loan application process significantly reduces the remaining processing time. However, this intervention results in a zero acceptance rate.

Therefore, the utility function defined in Equation 1 is instantiated for the two KPIs, as follows where $\bar{\lambda} \in [0, 1]$:

$$U_{\text{KPI}}(\sigma, (\bar{\lambda}, 1 - \bar{\lambda}), i) = \bar{\lambda}(1 - F_{\text{CO}}(\sigma, i)) + (1 - \bar{\lambda})F_{\text{RT}}(\sigma, i), \quad (5)$$

where F_{CO} represents the case outcome specific to the event log, as detailed in Section 5.1, returning 1 for a positive outcome and 0 for a negative one. Meanwhile, F_{RT} denotes the normalized remaining execution time obtained when performing (a, r) as the next step. It is worth noting that the combined outcome from Equation 5 is defined as a weighted linear combination of the two KPIs; however, these KPIs operate on different scales. Therefore, we scaled the remaining time values to the $[0, 1]$ range to ensure that they are numerically comparable with the binary case outcome. In particular, we transform the raw remaining time through three steps: first, we standardize it using *z-normalization*, then we apply *sigmoid transformation* and *min-max scaling* to map all values into the $[0, 1]$ interval. This three-step normalization procedure ensures that the originally skewed distribution of values is mapped to a nearly uniform range of $[0, 1]$, which allows for detecting even relatively small improvements in this perspective.

The prescriptive task aims to *minimize* the expected value of this combined KPI, as defined by the utility function U_{KPI} (5). In this setting, the parameter $\bar{\lambda}$ controls the trade-off between the two KPIs: higher values give more importance to the case outcome, while lower values place more emphasis on time efficiency. This flexibility allows stakeholders to align the system's behavior with their strategic goals. For instance, in a loan application process, a higher value of $\bar{\lambda}$ may be desirable to prioritize maximizing the proportion of applications that reach a successful outcome. Conversely, in a public administrative process, a lower $\bar{\lambda}$ may be preferred to emphasize reducing execution time, even if this comes at the cost of a lower customer satisfaction rate.

5.1 Case Studies

Our assessment is based on four real-life event logs, which are described below and in Table 1. The **Bank Account Closure (BAC)** is a process executed at a banking insti-

Table 1: Descriptive Statistics of Event Logs

Event Log	# cases	# activities	# resources	Mean case duration	% positive case
BAC	29194	15	654	16.4 days	82.5%
BPIC2017 - Before	12121	24	103	20.2 days	56.4%
BPIC2017 - After	14066	24	119	21.9 days	54.7%
BPIC2012	4685	23	67	15.1 days	47.9%

tution.⁷ The process involves closing customers’ accounts, which may be requested by the customer or the bank for several reasons. The **BPIC2012** [14] and **BPIC2017** [15] were published in relation to the Business Process Intelligence (BPIC) in 2012 and 2017, respectively. These datasets contain the execution history of a loan application process in the same Dutch financial institution. Due to a concept drift in the BPIC2017 log (cf. [2]), we partition the event log BPIC2017 into two sub-logs: one contains traces before week 22, namely **BPIC2017 - Before**, and the other contains those after week 28, namely **BPIC2017 - After**, allowing us to experiment across four distinct logs.

Furthermore, for each event log, the case outcome was encoded based on the occurrence of a predefined activity that marks a successful or unsuccessful outcome. Particularly, in the BAC case study, the activities “Network Adjustment Request” and “Back-Office Adjustment Request” are associated with unsuccessful outcomes, since they correspond to contingency actions, which should be avoided.⁸ For the remaining three case studies (BPIC2012, BPIC2017 - Before, BPIC2017 - After), we define a positive case outcome as the occurrence of the activity “O_Accepted”, which corresponds to customers accepting the offers proposed by the bank.

5.2 Experimental Setting

The event log $\mathcal{L}$ was temporally split into a training set $\mathcal{L}^{\text{train}}$ and a test set $\mathcal{L}^{\text{test}}$, where the temporal ordering of cases is defined by the timestamp of each case’s first event. To extract the training set, we compute the earliest time t_{split} in which 80% of the traces in $\mathcal{L}$ are completed, and the remaining 20% are used to define the test set. In particular, we keep all traces that are ongoing at t_{split} as $\mathcal{L}^{\text{test}}$ and exclude those starting after t_{split} . Traces that start after t_{split} are excluded from $\mathcal{L}^{\text{test}}$. The latter are removed because the evaluation is designed to reflect a realistic scenario in which recommendations are applied at t_{split} , leveraging only the information available up to that moment.

From $\mathcal{L}^{\text{test}}$ we create a test log $\mathcal{L}^{\text{run}}$ of running traces by truncating traces at the split time t_{split} , simulating ongoing, incomplete executions which allow us to evaluate the effectiveness of the recommendation module. $\mathcal{L}^{\text{train}}$ is used to train the predictive model and build the feasibility prescription module, while $\mathcal{L}^{\text{run}}$ and $\mathcal{L}^{\text{test}}$ are dedicated to assessing prescriptions.

In our experimental setting, the Genetic method was implemented using DiCE (Diverse Counterfactuals Explanations) [31], a model-agnostic library for generating di-

⁷ The BAC dataset is confidential and cannot be publicly shared.

⁸ To ensure a fair evaluation, we exclude from the set of activities that can be prescribed the key activity defining the case outcome.

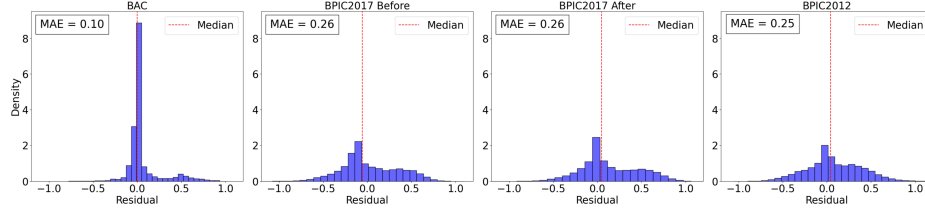


Fig. 3: Residual distributions and MAE values across the four datasets

verse counterfactual examples. In our case, we focus on generating alternative next activity-resource pairs in order to minimize the KPIs utility function U_{KPI} . Specifically, we configure the genetic algorithm (GA) with an initial population size of $p = 50$ and a maximum of 10 iterations, which have been found to be sufficient to explore the recommendation space while maintaining a reasonable computational cost. The threshold in the validity definition is set to $\tau = 0.05$ and the weighting factor of the *prox* measure, α , is set to $\alpha = 0.2$, ensuring a moderate contribution to the fitness score while still allowing different pairs to be explored. Moreover, we query the GA to return the top m solutions, where $m = 5$. Finally the set of feasible recommendations obtained from the feasibility prescription module, as described in Section 4.1, are obtained using a state abstraction with window size $w = 5$.

5.3 Evaluation of the Predictive Model

After preprocessing the data, as detailed in Section 4, we train the predictive model ϕ to estimate the multi-KPI utility function U_{KPI} . We instantiated ϕ by using a CatBoost model [16], due to its ability to achieve high predictive quality while maintaining efficient training time [19]. Additionally, we utilized a cross-validation approach to optimize the model’s hyperparameters (*learning_rate*, *depth*, *iterations*), ensuring robust generalization across different process instances.

Figure 3 reports on the mean absolute error (MAE) and the residual distributions of the predictive model over the four case studies in Section 5.1. The residual is defined as the difference between the actual combined KPI U_{KPI} and its predicted value when $\lambda = 0.5$. While the predictor for the BAC case study shows the highest accuracy with a small MAE and a tight residual distribution, the others are also satisfactory. The residual distributions show that the median value is close to zero, indicating that the predictive model does not systematically over- or under-predict. These results indicate that the prediction model is certainly still accurate enough to rank prescriptions.

5.4 Evaluating Prescriptions via Simulation

Ideally, we would perform an A/B test to compare the performance of process instances with and without following recommendations. However, in practice, this is infeasible because we cannot directly intervene in an ongoing process solely for evaluation purposes. Therefore, this study applies *business process simulation* to simulate the effect

of following and not following the recommendations under realistic conditions, as simulation is commonly used in PrPA for evaluating the effectiveness of recommendations [11].

The evaluation for each case study starts from the set of running cases $\mathcal{L}^{\text{run}} = \{\sigma'_1, \dots, \sigma'_n\}$, defined in Section 5.2. Each trace σ'_i is a prefix of a corresponding trace σ_i in $\mathcal{L}^{\text{test}}$, where σ_i can be seen as $\sigma'_i \oplus \langle (a'_i, t'_i, r'_i, \mathbf{d}'_i) \rangle \oplus \sigma''_i$.⁹ If the prescription is the pair (a, r) , we move the simulation to the state reached after performing $\sigma'_i \oplus \langle (a, t_{\text{split}}, r, \mathbf{d}) \rangle$, namely the prescription is assumed to occur at the simulation time t_{split} . We then simulate the continuation of the trace to determine how the trace would complete as a result of following the recommendation. As a *baseline* scenario, we assume no prescription is provided, and thus activity a'_i is performed by r'_i at time t'_i update the process' attribute as per $\mathbf{d}'_i$; at this stage, we let the simulation carry out the process instance. In each setting, each simulation is repeated ten times to mitigate the potential consequences of simulation stochasticity. Simulations were conducted using the ProSiT simulator [39].

The impact of assigning an activity–resource pair to a running trace is evaluated using the two KPIs considered in the optimization problem: the remaining execution time and the case outcome, obtained through simulation. The effectiveness of a prescription is measured as the improvement achieved by following the pairs (a, r) recommended by the Exhaustive and Genetic approaches, compared with the Ground Truth (GT) baseline, where no recommendation is applied. We compute the average improvement for the two KPIs on the set of running traces $\mathcal{L}^{\text{run}}$, as:

$$\begin{aligned} \Delta_{\text{CO}} &= \mathbb{E}_{\sigma' \in \mathcal{L}^{\text{run}}} (F_{\text{CO}}(\sigma_{\text{pres}}, |\sigma'|) - F_{\text{CO}}(\sigma_{\text{GT}}, |\sigma'|)), \\ \Delta_{\text{RT}} &= \frac{\mathbb{E}_{\sigma' \in \mathcal{L}^{\text{run}}} (F_{\text{RT}}(\sigma_{\text{GT}}, |\sigma'|) - F_{\text{RT}}(\sigma_{\text{pres}}, |\sigma'|))}{\mathbb{E}_{\sigma' \in \mathcal{L}^{\text{run}}} (F_{\text{RT}}(\sigma_{\text{GT}}, |\sigma'|))}, \end{aligned} \quad (6)$$

where $\mathbb{E}_{\sigma' \in \mathcal{L}^{\text{run}}}$ denotes the average over all prefixes, σ_{GT} is the simulated completion of the prefix σ' when no recommendation is applied, and σ_{pres} is the simulated completion when a prescription is applied. The latter depends on both the recommendation method (Exhaustive or Genetic) and the value of $\bar{\lambda}$ used in the utility function (5).

6 Results and Discussion

This section reports on the results obtained from the evaluation, emphasizing our answers to the two research questions formulated in Section 5.

We begin by examining how well the simulator reproduces the original process behavior. Figure 5 and Table 2 present the trace-duration distributions and proportions of positive traces, respectively, comparing the real and simulated event logs for each case study. These results show how well the simulation model captures the process perspectives relevant for the optimisation, supporting its suitability for evaluation purposes.

The answer to both research questions is based on Figure 4, which visualizes the performance improvement (6) achieved by the Exhaustive search and Genetic methods for the four case studies as well as for two KPIs used in the evaluation, namely remaining time and case outcome. Results are reported for different values of $\bar{\lambda}$ (cf. Equation 5)

⁹ Operator $\oplus$ is used to concatenate sequences

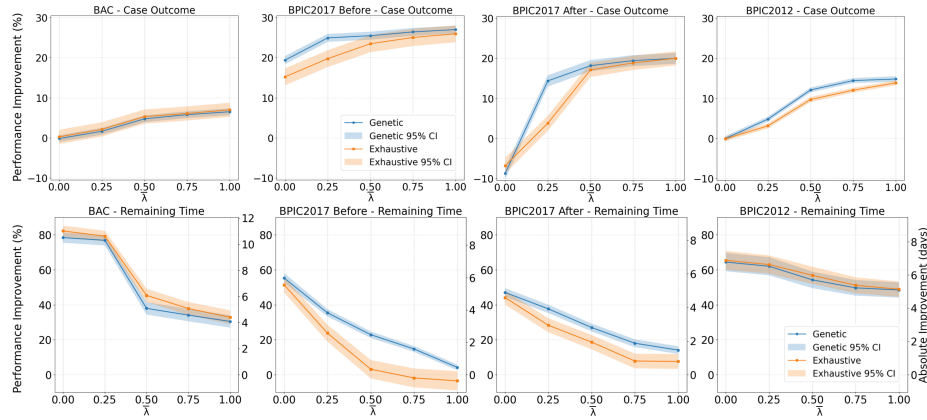


Fig. 4: Performance improvement achieved by the Genetic and Exhaustive approaches across multiple assignments of $\bar{\lambda}$, ranging from 0 to 1. For the remaining time, the right y-axis additionally indicates the absolute improvement in days.

to explicitly explore the trade-off between the two KPIs. The blue and orange lines refer to the average improvement for the Genetic and Exhaustive method, respectively, whereas the corresponding shaded bands indicate the 95% confidence interval.

6.1 Research Question RQ1

RQ1 focuses on comparing the improvements obtained from the Exhaustive search and Genetic methods. Figure 4 shows that the bands for the Genetic method are higher than or overlap the Exhaustive method, for every value of $\bar{\lambda}$ and for every use case. This illustrates that the Genetic method can trigger performance improvements that are as good as, or even better than, those of the Exhaustive method. A special reflection needs to be made for those case studies, such as in BPIC2017 Before, where the Genetic method outperforms the Exhaustive one, for the KPI related to the case outcome and/or remaining time, and for one or more $\bar{\lambda}$ values. This might seem counterintuitive at first, because the Genetic method explores a subset of solutions that the Exhaustive method does. This difference can be attributed to the different search strategies. The Exhaustive approach selects prescriptions solely based on the predicted KPI values, choosing the lowest. Given that for some pairs, especially infrequent pairs, the predictive model has seen little prior evidence, such predictions may be more error-prone. The error would typically lead to over- or under-estimating the actual KPI values, thus changing the ranking of the prescriptions, and ultimately causing to not choose the actual, best prescription. Our hypothesis is that, conversely, the Genetic approach, by considering the *proximity* to the most probable activity-resource pair observed in the training log (Section 4.3), tends to favor more frequently observed pairs, which are expected to provide more robust estimations, thus returning a more accurate prescription ranking.

In summary, *the Genetic method performs at least as well as the exhaustive method*, in every setting that was tested. This is a particularly remarkable result because the Ge-

Table 2: Rate of positive outcome of real and simulated log

Case Study	Real log	Simulated log
BAC	82.5%	81.3%
BPIC2017 - Before	56.4%	53%
BPIC2017 - After	54.7%	54.6%
BPIC2012	47.9%	46.9%

Table 3: Execution Time of Recommendation Strategies

Case Study	Genetic	Exhaustive
BAC	0.51s	0.6s
BPIC2017 - Before	0.42s	0.54s
BPIC2017 - After	0.37s	0.53s
BPIC2012	0.3s	0.43s

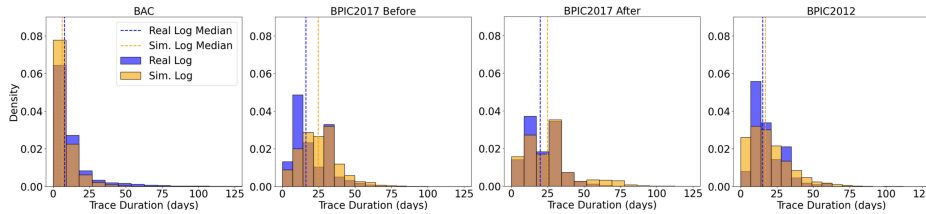


Fig. 5: Residual distribution for simulator performance.

netic method is significantly faster, as it only explores a subset of the possible solutions. Table 3 shows the computation time for different case studies, grouping by the different values of $\bar{\lambda}$: the average computation time for the Genetic and Exhaustive methods is 0.4 and 0.525 seconds over a single trace, respectively. It thus follows that the *Genetic method leads to a 23.8% reduction in computation time.*

6.2 Research Question RQ2

This question is answered by observing the performance improvements shown in Figure 4 for increasing values of $\bar{\lambda}$. Larger values of $\bar{\lambda}$ provide more importance for improving the KPI related to case outcome, and less importance for reducing the remaining time. As expected, larger values of $\bar{\lambda}$ generally increase the performance improvement for outcome-related KPI and reduce the improvement for the KPI linked to the remaining time. For $\bar{\lambda} = 0.5$, we typically observe that the performance improvements for the two KPIs are located between the maximum and minimum, which are conversely obtained for $\bar{\lambda} = 0$ and 1. This is also expected because intermediate values are meant to indicate that we mediate the improvement of the involved KPIs.

Interesting results can be observed for $\bar{\lambda} = 1$, when no improvement is expected related to the remaining time. However, in some particular cases (more specifically, for the BAC and BPIC2012 use cases), we still observe a significant improvement for the remaining time. This improvement is related to the observation that, for the BAC case study, the positive cases exhibit a significantly shorter average duration, almost 50% (12 days) less than the negative cases (25 days). This suggests a correlation between decreasing in remaining time and improving case outcome. The same correlation was observed - although less marked - for the BPIC2012 case study: positive cases have a 20% lower average duration (16 days) compared to the negative cases (21 days).

6.3 Threat of Validity and Future Directions

This section reflects on the limitations of the proposed framework and outlines directions for future research.

A first limitation concerns the trade-off weight λ_i . Although experiments indicate that intermediate values of λ_i tend to yield balanced improvements across KPIs, the framework currently relies on user-specified preferences. An adaptive, data-driven procedure for tuning or learning λ_i remains an open problem and would help streamline the approach more in settings where KPI balance may fluctuate over time.

A potential limitation of the present work concerns the scope of the recommendation task. The framework focuses on prescribing the next step for a single running trace at a time. This choice reflects the goal of establishing a general machinery for multi-objective optimization, but it limits the evaluation to isolated recommendations. In many operational settings, multiple traces progress concurrently, and prescriptions for one instance may interact with or constrain those for others. This is far from being trivial because it would require additional mechanisms for coordinating recommendations across multiple instances: this was the focus of the work by Padella et al. [32], which we plan to eventually integrate.

Finally, although the framework recommends only the next step, such interventions influence the full continuation of the trace during simulation. While this aligns with the idea that local guidance propagates downstream, it remains unclear how next-step and full continuation optimization compare in practice. Studying whether direct suffix-level prescriptions yield additional benefits is a natural direction for future work.

7 Conclusions

Existing approaches for Prescriptive Process Analytics (PrPA) typically focus on improving a single Key Performance Indicator (KPI), whereas real-world process performance is often measured on multiple interdependent, and even conflicting, KPIs.

This work proposes a multi-objective framework for prescriptive process analytics that explicitly models and navigates trade-offs among competing KPIs. Recommendations are framed as activity-resource pairs, indicating which activity should be executed next and to whom it should be allocated. Since this formulation produces a large search space to navigate through, this paper also proposes a genetic method that efficiently explores activity-resource combinations.

Experiments over four use cases demonstrate the viability of multi-objective PrPA for balancing multiple KPIs and analyze how parameter tuning affects the trade-offs. Results also show that the genetic method can provide a similar degree of KPI improvements while saving 23.4% of computation time.

Acknowledgments. This work has been partially funded by the PNRR project FAIR - Future AI Research (PE00000013), including the project “Future AI Research (FAIR) - Spoke 2 Integrative AI - Symbolic conditioning of Graph Generative Models (SymboliG)”. Ngoc-Diem Le is financially supported by MUR (PNRR) and the University of Padua.

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. Agostinelli, S., Benvenuti, D., Casciani, A., De Luzi, F., Marinacci, M., Marrella, A., Rossi, J.: A context-aware framework to support decision-making in production planning. In: *Proceedings of the International Conference on Advanced Information Systems Engineering*. pp. 248–264. Springer (2024)
4. 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. IEEE (2021)
5. Bozorgi, Z.D., Teinemaa, I., Dumas, M., Rosa, M.L., Polyvyanyy, A.: Prescriptive process monitoring based on causal effect estimation. *Inf. Syst.* **116**, 102198 (2023)
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)*. vol. 458, pp. 137–154. Springer (2022)
7. Buliga, A., Di Francescomarino, C., Ghidini, C., Donadello, I., Maggi, F.M.: Guiding the generation of counterfactual explanations through temporal background knowledge for predictive process monitoring. *Data Mining and Knowledge Discovery* **39**(5), 1–49 (2025)
8. Buliga, A., Meneghello, F., Graziosi, R., Ronzani, M.: Enhanced what-if scenarios generation by bridging generative models and process simulation. In: *Proceedings of the 7th International Conference on Process Mining (ICPM 2025)*. pp. 1–8. IEEE (2025)
9. Chapela-Campa, D., Dumas, M.: Enhancing business process simulation models with extraneous activity delays. *Information Systems* **122**, 102346 (2024)
10. de Leoni, M., Dees, M., Reulink, L.: Design and evaluation of a process-aware recommender system based on prescriptive analytics. In: *Proceedings of the 2nd International Conference on Process Mining (ICPM 2020)*. pp. 9–16. IEEE (2020)
11. De Moor, J., Weytjens, H., De Smedt, J., De Weerd, J.: Simbank: from simulation to solution in prescriptive process monitoring. In: *Proceedings of the 23rd International Conference on Business Process Management (BPM 2025)*. pp. 165–182. Springer (2025)
12. Di Francescomarino, C., Ghidini, C.: Predictive process monitoring. In: *Process Mining Handbook*, pp. 320–346. LNBIP, Springer International Publishing, Berlin-Heidelberg (2022)
13. 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)
14. van Dongen, B.: Bpi challenge 2012 (2012). <https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f>
15. van Dongen, B.: Bpi challenge 2017 (2017). <https://doi.org/https://doi.org/10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b>
16. 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)
17. Dumas, M., Rosa, L.M., Mendling, J., Reijers, A.H.: *Fundamentals of business process management*. Springer (2018)
18. 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** (2022)

19. 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)
20. Gröger, C., Schwarz, H., Mitschang, B.: Prescriptive analytics for recommendation-based business process optimization. In: *Proceedings of the International Conference on Business Information Systems*. pp. 25–37. Springer (2014)
21. Kis, I., Bachhofner, S., Di Ciccio, C., Mendling, J.: Towards a data-driven framework for measuring process performance. In: *Proceedings of the International Workshop on Business Process Modeling, Development and Support*. pp. 3–18. Springer (2017)
22. Kubrak, K., Botchorishvili, L., Milani, F., Nolte, A., Dumas, M.: Explanatory capabilities of large language models in prescriptive process monitoring. In: *Proceedings of the 22nd International Conference on Business Process Management (BPM 2024)*. pp. 403–420. Springer Nature Switzerland, Cham (2024)
23. Kubrak, K., Milani, F., Nolte, A., Dumas, M.: Prescriptive process monitoring: Quo vadis? *PeerJ Comput. Sci.* **8**, e1097 (2022)
24. Le, N.D., Padella, A., Vinci, F., De Leoni, M.: Leveraging counterfactuals for prescriptive process analytics. In: *Proceedings of the 23rd International Conference on Business Process Management (BPM 2025)*. pp. 200–215. Springer (2025)
25. Marler, R.T., Arora, J.S.: Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization* **26**(6), 369–395 (2004)
26. Márquez-Chamorro, A.E., Resinas, M., Ruiz-Cortés, A.: Predictive monitoring of business processes: A survey. *IEEE Transaction on Services Computing* **11**(6) (2018)
27. Meneghello, F., Middelhuis, J., Genga, L., Bukhsh, Z., Ronzani, M., Francescomarino, C.D., Ghidini, C., Dijkman, R.M.: Optimizing resource allocation policies in real-world business processes using hybrid process simulation and deep reinforcement learning. In: *Proceedings of the 22nd International Conference on Business Process Management (BPM 2024)*. vol. 14940, pp. 167–184. Springer (2024)
28. Metzger, A., Kley, T., Palm, A.: Triggering proactive business process adaptations via online reinforcement learning. In: *Proceedings of the 18th International Conference on Business Process Management (BPM 2020)*. pp. 273–290. Springer (2020)
29. Metzger, A., Neubauer, A., Bohn, P., Pohl, K.: Proactive process adaptation using deep learning ensembles. In: *Proceedings of the International Conference on Advanced Information Systems Engineering*. pp. 547–562. Springer (2019)
30. Mitchell, M.: *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA (1998)
31. 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)
32. 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)
33. 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)* (2022)
34. 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)*. pp. 345–362. Springer Nature Switzerland, Cham (2024)
35. Shoush, M., Dumas, M.: When to intervene? prescriptive process monitoring under uncertainty and resource constraints. In: *Proceedings of the 20th International Conference on Business Process Management (BPM 2022)* (2022)

36. Shoush, M., Dumas, M.: Prescriptive process monitoring under resource constraints: A reinforcement learning approach. *KI - Künstliche Intelligenz* (2024)
37. Shoush, M., Dumas, M.: White box specification of intervention policies for prescriptive process monitoring. *Data and Knowledge Engineering* **155**, 102379 (2025)
38. Teinemaa, I., Tax, N., de Leoni, M., Dumas, M., Maggi, F.M.: Alarm-based prescriptive process monitoring. In: *Proceedings of the 16th International Conference on Business Process Management (BPM 2018)*. pp. 91–107. Springer (2018)
39. Vinci, F., Park, G., van der Aalst, W., de Leoni, M.: Prosit: A tool for interactive and transparent process simulations. In: *Proceedings of the Workshops and Satellite Events of the 23rd International Conference on Service-Oriented Computing (ICSOC 2025)*. Springer (2025), in Press
40. Vinci, F., Park, G., Van Der Aalst, W.M., De Leoni, M.: Online discovery of simulation models for evolving business processes. In: *Proceedings of the 23rd International Conference on Business Process Management (BPM 2025)*. pp. 451–468. Springer (2025)
41. Weinzierl, S., Dunzer, S., Zilker, S., Matzner, M.: Prescriptive business process monitoring for recommending next best actions. In: *Proceedings of the 18th International Conference on Business Process Management (BPM 2020)*. pp. 193–209. Springer (2020)