Resource Allocation in Recommender Systems for Global KPI Improvement

Alessandro Padella(✉) and Massimiliano de Leoni

University of Padua, Padua, Italy
alessandro.padella@phd.unipd.it, deleoni@math.unipd.it

Abstract. Process-aware Recommender systems are information systems designed to monitor the execution of processes, predict their outcomes, and suggest effective interventions to achieve better results, with respect to reference KPIs (Key Performance Indicators). Interventions typically consist of suggesting an activity to be assigned to a certain resource. State of the art typically proposes interventions for single cases in isolation. However, since resources are shared among cases, this might impact the effectiveness of the available interventions for other cases that would require one. As result, the overall KPI improvement is partially hampered. This paper proposes an approach to assign resources to needed cases, aiming to improve the overall KPI values for all cases together, namely the summation of KPI values for all cases. Experiments conducted on two real-life case studies illustrate that globally considering all needing cases together allows a better global KPI improvement, compared with a more greedy approach where interventions are proposed one after the other.

Keywords: Process Improvement · Process Prescriptive Analytics · Recommender Systems · Resource Allocation · Resource Experience

1 Introduction

Process-aware Recommender Systems are a class of information system that aims to monitor whether executions are predicting to achieve the expected goals, and, whenever this is not the case, they propose interventions to try to take those executions back on track. In literature interventions are typically based on advising what activity to perform as next, possibly paired with a suggestion of the resource that will carry it out (see Sect. 2).

However, resources are shared among all running process instances, a.k.a. cases, and typically they can carry on one activity a time. As a result, if the interventions (activity and resource) are determined for each instance without considering the other instances that also require intervention, the overall effectiveness, namely for all instances that require interventions, is limited. For instance, if one process instance P1 is assigned a resource R1, R1 cannot work a different instance P2 that requires intervention. It might be the case that it

C. Di Francescomarino et al. (Eds.): BPM 2023 Forum, LNBIP 490, pp. 249–266, 2023.
https://doi.org/10.1007/978-3-031-41623-1_15

is more beneficial to assign a different resource R2 to P1, because R2 can work almost as good for P1, and let R1 work on P2, for which no resource exists that is almost equally good. This consideration illustrates how the decision of the interventions is a global decision for all running cases.

Section 2 reports how literature does not propose approaches for resource allocations to cases with the aim of improving the whole set of process instances. This paper proposes an approach to overtake this limitation. In our paper, the achievement of the goal is measured through a measurable Key Performance Indicator (KPI): cases associated with values outside the acceptable range are considered worth of an intervention. The problem that we tackle is clearly an optimization problem: given the likely process' scenario of hundreds of resources and running cases, an exact solution is practically unfeasible. We thus propose two greedy approaches, one faster and one slower, that respectively provide a worse and better approximation.

In a nutshell, the idea is to create an initial resource profile, in which resources are allocated to cases: each profile is associated with an expected overall KPI improvement. The expected overall KPI improvement is computed, using machine-learning techniques for prescriptive process analytics. Then, this initial profile is altered, by changing resources allocated to cases, thus obtaining further profiles. At the end, a number of resource profiles is generated, from which those with higher expected KPI improvements are retained. The ultimate outcome is a set of resource profiles, which are deemed valid to improve KPIs. In different resource profiles, the same resource is assigned to a different case and intervention: resources are thus given a certain degree of freedom on which case to pick up and work on as next. This is beneficial, because a rigid resource-to-case imposition is against the principle of resource-aware recommender systems [1], such as how the problem of task-to-resource assignments is typically tackled in operation research.

The framework has been assessed on real-life case studies with processes with hundreds of running cases and resources. This allowed us to perform a stress test on the practical feasibility. The typical behavior of resources was simulated, on the basis of behavior patterns observed in human-computer interaction literature. The results show that our resource-allocation framework enables a significantly higher total KPI improvements, i.e. considering all running cases, if compared with scenarions in which each case is recommended in isolation.

Section 2 discusses related works in the domain of prescriptive process analytics and resource-aware recommender systems. Section 3 introduces the necessary background concepts: event logs, KPI definitions, and process prescriptive analytics that consider single cases in isolation. Section 4 puts forward our approach for resource allocation for global KPI improvement, Sect. 5 reports on the evaluation setup and results, while Sect. 6 concludes the paper, summarizing the contribution in our paper.

2 Related Works

Literature has focused on using recommender systems in business processes to improve the future outcome of process instances. This has often been translated into being focused on recommending which activities to work on next to improve the process' Key Performance Indicators (KPIs) [2–4].

The growing interest in recommender systems for process mining has led the community to explore how to determine when to intervene with recommendations [5] and whether an intervention is cost-wise worth [6]. A body of research has also focused on ensuring that recommendation are well explained to human-resources [7], using Shapley Values theory [8].

Moreover, several research works have focused on considering which resources should perform specific activities in various contexts. Cabanillas et al. in [9, 10] propose two approaches to define a language to equip BPMN models with complex resource-allocation policies, as well as to discover those policies.

A few works focuses on suggesting a resource allocation for a set of activities that need to be performed, without - though - focusing on recommending which activity to perform. Zhao et al. in [11] provide a framework based on a system of convex equations that encode a system of constraints on time and cost. Huang et al. in [12] leverage on Reinforcement Learning to propose a resource-allocation algorithm based on a Markov decision process. Park et al. in [13] integrate offline prediction model construction, using Long Short-Term Memory models to predict the next activity to perform and, subsequently, employing a minimum-cost-and-maximum-flow algorithm to allocate resources. Dumas et al. in [14] focus on recommending resource-activity pair, as we aim here. However, they recommend for single cases in isolation, which was previously mentioned to provide a lower degree of overall KPI improvement, namely for the whole set of running cases.

In conclusion, no previous research works have pursued the goal to provide an global KPI improvements, while leaving a certain degree of freedom to resources on which case (and activity) to work on as next.

3 Preliminaries

The starting point for a process mining-based system is an *event log*. An event log is a multiset of *traces*. Each trace is a sequence of events, each describing the life-cycle of a particular *process instance* (i.e. a *case*) in terms of the *activities* executed, *resources* that execute it, and the process *attributes* manipulated.

Definition 1 (Events). *Let $\mathcal{A}$ be the set of process activities. 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 $(a, r, v) \in \mathcal{A} \times \mathcal{R} \times (\mathcal{V} \nrightarrow \overline{\mathcal{W}})$ where a is the event activity, r the resource that performs it and v is a partial function assigning values to process attributes with $v(x) \in \mathcal{W}_{\mathcal{V}}(x)$.*

A trace is a sequence of events. The same event can occur in different traces, namely attributes are given the same assignment in different traces. This means that the entire same trace can appear multiple times and motivates why an event log is to be defined as a function which assigns a trace to a given identifier:

Definition 2 (Traces & Event Logs). *Let $\mathcal{E} = \mathcal{A} \times \mathcal{R} \times (\mathcal{V} \nrightarrow \overline{\mathcal{W}})$ be the universe of events. Let $\mathcal{I}$ be the universe of the case identifiers. A trace σ a sequence of events, i.e. $\sigma \in \mathcal{E}^*$. An event-log $\mathcal{L}$ is here modeled as a function that, given an identifier i of a log trace returns the sequence of events related to the process instance with the identifier i, i.e. $\mathcal{L} : \mathcal{I} \rightarrow \mathcal{E}^*$.*[1]

Given an event $e = (a, r, v)$, the remainder uses the following shortcuts: $activity(e) = a$, $resource(e) = r$ and $variables(e) = v$. Also, given a trace $\sigma = \langle e_1, \ldots, e_n \rangle$, $prefix(\sigma)$ denotes the set of all prefixes of σ, including σ, namely $prefix(\sigma) = \{\langle\rangle, \langle e_1 \rangle, \langle e_1, e_2 \rangle, \ldots, \langle e_1, \ldots, e_n \rangle\}$.

For building our recommender system, we need to define what we aim to optimize, i.e. the goal of our recommendation: hereafter, this is named Key Performance Indicator (KPI) and depends on the specific process domain.

Definition 3 (KPI Function). *Let $\mathcal{E}$ be the universe of events. A Key Performance Indicator (KPI) is a function $\mathcal{K} : \mathcal{E}^* \times \mathbb{N} \rightarrow \mathbb{R}$ such that, given a (prefix of a) trace $\sigma \in \mathcal{E}^*$ and an integer $1 \leq i \leq |\sigma|$,*[2] *$\mathcal{K}(\sigma, i)$ returns the KPI value of σ after the occurrence of the first i events.*

Therefore $img(\mathcal{K})$ is the set of all possible KPI values. With abuse of notation, we indicate $\mathcal{K}(\sigma) = \mathcal{K}(\sigma, |\sigma|)$, namely the KPI value after the occurrence of events in trace σ. Note that our KPI definition is assumed to be computed a posteriori when the execution is completed and leaves a complete trail as a certain trace σ. In many cases, the KPI value is updated after the occurrence of each event, i.e. after each activity execution. We aim to be generic and account for all relevant domains. Given a trace $\sigma = \langle e_1, \ldots, e_n \rangle$ that records a complete process execution, the followings are examples of two potential KPI definitions:

- *Total Time.* We opted to consider the task in which the objective is to reduce the total time. Given a σ's prefix of i events, $\mathcal{K}_{total}(\sigma, i)$ measures the difference between the timestamp of the trace's last future event and the first event's timestamp.
- *Activity Occurrence.* It measures if a certain activity is going to occur in the future, such as an activity eventually *Open Loan* in a loan-application process. The corresponding KPI definition for the occurrence of an activity a is $\mathcal{K}_{occur\_a}(\sigma, i)$, which is equal to 1 if the activity a occurs in $\langle e_{i+1}, \ldots, e_n \rangle$, 0 otherwise.

[1] The operator * refers to the Kleene star: given a set A, A^* contains all the possible finite sequences of elements belonging to A.

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

Table 1. Example of the output of the Prescriptive Analytics Oracle Function in a tabular form, for a given trace when the KPI is the total time of a case. It provides the recommended activity *Back-Office Adjustment Requested* associated with a set of pairs of resources and delta in KPI. For instance, if the resource *BOCSER* executes an adjustment to the Back-Office, the expected total time of the procedure will decrease by 195 h.

Activity	Resource	Δ
Back-Office Adjustment Requested	*CE_UO*	208 h
	BOCSER	195 h
	BOC	112 h
	...	...

The goal of the recommender system is to provide recommendations on both the activities to be performed and the resources best suited to perform them, with the aim of enhancing the final outcome of running process instances in terms of the identified KPIs. To achieve this, a **Prescriptive Analytics Oracle Function** must be developed. This function will enable the prediction of the KPIs of the final outcome of a running process instance, and will identify the best activity to be performed and the most suitable resource to perform it.

Definition 4 (Prescriptive Analytics Oracle Function). *Let $\mathcal{E}$ be the universe of events and $\sigma \in \mathcal{E}^*$ a (running) trace belonging to it, $\mathcal{A}$ the set of possible activities, $\mathcal{K} : \mathcal{E}^* \times \mathbb{N} \to \mathbb{R}$ a KPI function and $\mathcal{R}$ the set of the possible resources. A Prescriptive Analytics Oracle Function is a function $\psi : \mathcal{E}^* \times \mathcal{K} \to \mathcal{A} \times 2^{(\mathcal{R} \times \mathbb{R})}$ such that $\psi(\sigma, \mathcal{K})$ returns $(a, \{(r_1, \Delta_1), \ldots, (r_m, \Delta_m)\})$ with $m \leq |\mathcal{R}|$ to indicate that activity a is recommended and, if performed by r_i, will lead to a Δ_i improvement of KPI $\mathcal{K}$. Also, $\forall i, j \in \{1, \ldots, m\}$, $r_i = r_j \iff i = j$, meaning that a resource can only be recommended once.*

Since not all resources can perform the recommended activity a, the number m of recommended resources does not necessarily coincide with the number $|\mathcal{R}|$ of all resources [15]: also, some resources might not be available at a certain point for other reasons, e.g. on holidays or on sick leave.

In the example in Table 1, the oracle function ψ takes as input the KPI function as defined in Definition 3, with $\mathcal{K}$ modelling the KPI . For a certain trace, the recommended activity is *Back-office Adjustament Requested*. The function also returns a set of pairs (r, Δ) to indicate that, if the activity is performed by resource r, the final KPI is predicted to change by Δ. Concretely, if the activity *Back-Office Adjustment Request* is performed, e.g., by `CE_UO`, the total time will reduce by `208hours`.

In the remainder, we will make use of a helper function $max_{\psi,\mathcal{K}}(\sigma)$ that for each (running) trace $\sigma \in \mathcal{E}^*$ returns the maximum achievable improvement.

Definition 5 (Helper function). *Let $\mathcal{E}$ be the universe of events. $\sigma \in \mathcal{E}^*$ a (running) trace belonging to it, and $\mathcal{A}$ the set of possible activities. Let $\psi(\sigma, \mathcal{K}) = (a, \{(r_1, \Delta_1), \ldots, (r_m, \Delta_m)\})$ the Prescriptive Oracle Function. The Helper function $max_{\psi,\mathcal{K}} : \mathcal{E}^* \rightarrow \mathcal{A} \times \mathbb{R}$ is a function that returns a pair $(a, \Delta) = (a, max(\{\Delta_1, \ldots, \Delta_m\}))$*

The oracle function can be implemented in multiple ways, using several of the prescriptive-analytics algorithms in literature (cf. Sect. 2). This paper does not aim to propose any specific prescriptive-analytics algorithms. However, for the implementation and testing, we opted to use the prescriptive-analytics proposal discussed in [7], which has been extended to also return the pairs of resources and KPI's deltas.[3]

4 Global Activity-Resource Allocation

The purpose of this paper is to provide resources with tailored recommendations regarding which actions to take and to which process instance, while also allowing for a degree of choice and autonomy. To achieve this goal, it is essential to establish a framework that can generate interdependent recommendations while simultaneously accommodating the individual decision-making processes of the resources involved. It is important to note that the best recommendation for a case, when viewed from the perspective of optimising a specific KPI, may not necessarily be the best recommendation for that case in the context of global optimisation. In the task of our work, the KPI to be optimised is not pointwise: an individual case may get the best recommendation for him, but this makes a resource busy and so unavailable for other cases that could improve their KPI more. Hence, we want to optimise the sum of the KPIs for all resources and cases on which they act, ensuring the single recommendations provided to resources interact with each other without conflict. This leads to the definition of a **Profile**.

Definition 6 (Profile). *Let $\mathcal{L} : \mathcal{I} \rightarrow \mathcal{E}^*$ be an event log, $\mathcal{A}$ the set of its possible activities, and $\mathcal{R}$ the set of the possible resources. A profile $\mathcal{P} \subset (\mathcal{I} \times \mathcal{A} \times \mathcal{R} \times \mathbb{R})$ is defined as a set of tuples (i, a, r, Δ) where for the (running) case with identifier i, the activity a is to be assigned to the resource r for improving its expected KPI of Δ. There is an additional constraints: there cannot be two tuples with the same case identifier or the same resource.*

A profile aims to allocate the set of available resources to a set of cases with the aim of improving the overall KPI values for all running cases. The generation of a profile is challenging: it is a combinatorial problem that would require one to potentially try every combination of case ids, activities, and resources. This is practically unfeasible. In Sect. 4.1, we illustrate a greedy algorithm to compute a profile.

[3] Code available at https://github.com/Pado123/prescriptive_global_optimization.

Id	Activity	ΔKPI
DD-45678	Pending Liquidation Request	93434 h
CC-34567	Back-Office Adjustment Requested	85014 h
BB-23456	Pending Liquidation Request	42543 h
…	…	…

Id	Activity	Resource	ΔKPI
DD-45678	Pending Liquidation Request	BOCSER	93434 h
BB-23456	Pending Liquidation Request	BOC	21944 h
CC-34567	Back-Office Adjustment Requested	CE_UO	10433 h
…	…	…	…

Fig. 1. The table on the left is an example of tabular form of the $\Delta RANK$ sequence: the columns shows the case ids, the recommended activities for the respective cases, and the maximum KPI improvement. In this example, the employed KPI is the case total time. The profile is obtained from $\Delta RANK$ by allocating resources to cases (see the right-hand side table). Since the best resource cannot be assigned to every case, the assigned resource might cause a drop in the KPI's improvements.

The creation of a single profile is also poorly applicable in practice because it would impose activities to resources, without considering external factors. The novelty of our framework is also linked to providing process actors with some degree of freedom, while still aiming to improve the overall KPI. This requires generating several profiles: different profiles assign different resources to a certain resource. A resource can pick one of the activities available for him/her in any of the generated profiles.

The resource's choice naturally filters out profiles that are incompatible with the choice made. The subsequent resource to choose will then have fewer profiles according to which to choose. Section 4.2 illustrates how to generate the profiles additional to the first.

4.1 Generation of the First Profile

To create an initial profile $\mathcal{P}_0$, we first create a sequence $\Delta RANK \subseteq (\mathcal{I} \times \mathcal{A} \times \mathbb{R})^*$. It can be constructed using the Helper function defined in Definition 5 as follows. First, we build the set of triples $(i_1, a_1, \Delta_1), \ldots, (i_n, a_n, \Delta_n) = \bigcup_{i \in dom(\mathcal{L})} (i, max_{\psi,\mathcal{K}}(\mathcal{L}(i))$, which later are sorted descending by the third component, namely $\Delta_1, \ldots, \Delta_n$. An example of $\Delta RANK$ is given in the left-hand side table in Fig. 1.

The first profile $\mathcal{P}_0$ is obtained by extending $\Delta RANK$ with resources (cf. the right-hand side table in Fig. 1). To achieve this, we start from the first element $(i_1, a_1, \Delta_1) \in \Delta RANK$, i.e. the one with the greatest expected improvement. Then, we evaluate $\psi(\mathcal{L}(i_1), \mathcal{K}) = (a_1, \{(r_1^1, \Delta_1^1), \ldots, (r_1^m, \Delta_1^m\}))$, with $\mathcal{K}$ be the KPI function of interest, and we associate resource r_1^1 to (i_1, a_1, Δ_1) the first pair (r_1, Δ_1), thus resulting to add $(i_1, a_1, r_1, \Delta_1^1)$ to the profile. Resource r_1 is removed from the set $\mathcal{R}$ of the resources available.

We then move to the second element $(i_2, a_2, \Delta_2) \in \Delta RANK$, and evaluate $\psi(\mathcal{L}(i_2), \mathcal{K}) = (a_2, \{(r_2^1, \Delta_2^1), \ldots, (r_2^q, \Delta_2^q\}))$. If $\{r_2^1, \ldots, r_2^q\} \cap \mathcal{R} = \emptyset$, no element is added to profile $\mathcal{P}_0$ for instance i_2. Otherwise, we look for the smallest j such

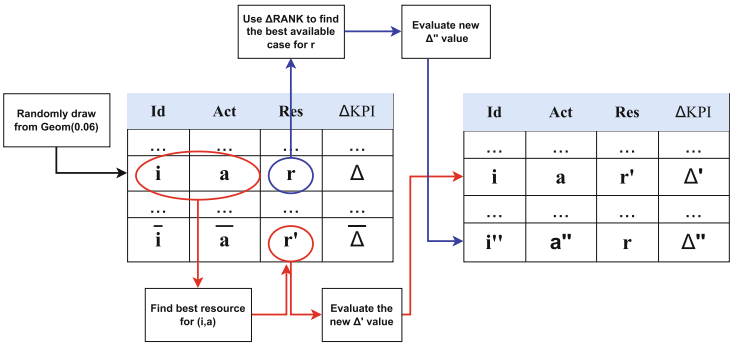

Fig. 2. A visual representation of the algorithm to perturb profiles. The right table depicts the original profile, while the left table shows the perturbed profile. A random element $p = (i, a, r, \Delta)$, according to a geometric distribution. The algorithm identifies the best new resource r' for the corresponding trace identifier i and activity a, resulting in a new element $p' = (i, a, r', \Delta')$ (indicated in red in the picture). The resource r' is unassigned from the previous assignment: the element for r' is thus removed from the profile ($\overline{i}, \overline{a}, r', \overline{\delta}$ in figure). Resource r is free, and is given a different assignment (element i'', a'', r, Δ'' in figure). (Color figure online)

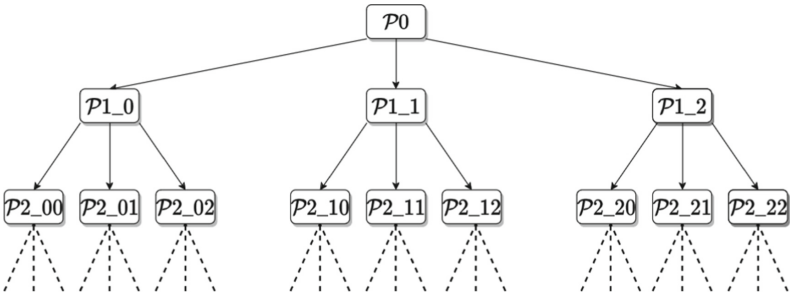

Fig. 3. Schematic of how profiles are generated via perturbation. Starting from the initial profile $\mathcal{P}_0$, a number of perturbations are created (three for the example in figure), namely: $\mathcal{P}_{1\_0}$, $\mathcal{P}_{1\_1}$ and $\mathcal{P}_{1\_2}$. This is then repeated for each of the obtained profiles, until a certain number of profiles are overall constructed. If a perturbation generates a profile that has already been created, this is discarded.

that $r_2^j \in \mathcal{R}$. Tuple $(i_2, a_2, r_2^j, \Delta_2^j)$ is added to profile $\mathcal{P}_0$. Note that Δ_2^j might be lower that Δ_2 because the allocated resource might not yield the maximum improvement: by construction, it is only guaranteed that $\Delta_2 = \Delta_2^1$. Resource r_2^j is removed from $\mathcal{R}$. This procedure is repeated for every tuple in $\Delta RANK$, as long as set $\mathcal{R}$ is not empty (i.e. activities and cases can be allocated to resources).

4.2 Generation of Additional Profiles

The first profile $\mathcal{P}_0$ is certainly the valuable starting point, but it falls short in two main aspects. It is generated considering the traces in the descending order of potential improvements: it is in fact a greedy approach, which might still returned solutions relatively far from the potential, optimal solution. Using approaches based on local search, solution $\mathcal{P}_0$ is perturbed to obtain more solutions of profiles. As discussed, we want to grant freedom to resources on the choice of which cases (and consequently activities) to work on as next: therefore, all profiles generated by perturbation are retained.

Figure 2 illustrates how one profile $\mathcal{P}$ is perturbed into $\mathcal{P}'$: elements are visualized in tabular form. Initially $\mathcal{P}' = \mathcal{P}$. The elements in $\mathcal{P}$ are sorted by descending values of the KPI improvement (see column ΔKPI in figure). An element $p = (i, a, r, \Delta)$ is randomly selected from the sorted list according to a geometric distribution with $p = 0.06$. Let $p = (i, a, r, \Delta)$ the selected element, with oracle function $\psi(\mathcal{L}(i), K) = (a, \{(r_1, \Delta_1), \ldots, (r_m, \Delta_m\})$. Elem p is removed from $\mathcal{P}'$, while we add a certain $p' = (i, a, r', \Delta')$ such that, if $r = r_1$, then $r' = r_2$ an $\Delta' = \Delta_2$, otherwise $r' = r_1$ and $\Delta' = \Delta_1$. Since every resource is assigned to the activity of some case, $\mathcal{P}$ contains some element for r': $\overline{p} = (\overline{i}, \overline{a}, r', \overline{\Delta}) \in \mathcal{P}$. Tuple $\overline{p}$ is removed from $\mathcal{P}'$. Resource r is now free: we pick the top element $(i'', a'', \Delta'') \in \Delta RANK$ such that r is allowed to execute a'' and there is no element in the $\mathcal{P}'$ that refers to the case with id i''. Element (i'', a'', r, Δ'') is added to $\mathcal{P}'$.

In sum, the above procedure is able to perturb a profile and, thus, create a new one. This is iterated, until a given target of profiles is created. This can be visualized as in Fig. 3: we started from the initial profile $\mathcal{P}_0$. A certain number of perturbations is created from $\mathcal{P}_0$: profiles $\mathcal{P}_{1\_0}$, $\mathcal{P}_{1\_1}$ and $\mathcal{P}_{1\_2}$ in figure with three perturbations. This is then repeated for each of the obtained profiles. In general, two subsequent perturbations can result in the original profiles; however, we discard the perturbed profiles that were already previously obtained. This motivates why Fig. 3 has a tree-like structure, in place of a graph-like.

4.3 Assign Recommendations

Once we generate the entire set of profiles, we create a Profiles Ranking $\mathscr{P}$, by sorting them down by global KPI improvements (i.e., summing up the KPI improvements for all elements in the profiles).

which is used to effectively provide recommendations to resources. In fact, in organizational reality, they receive the range of choices provided for them based on the profile's order in which the Profiles Ranking $\mathscr{P}$ has been sorted. At the end of the assignment procedure, every resource r will have selected an activity a and a case identifier i, resulting in a final set that, from this point onwards, we will refer to it as **Resource-task Assignment Set** $\mathcal{S} \subset (\mathcal{I} \times \mathcal{A} \times \mathcal{R} \times \mathbb{R})^{|\mathcal{R}|}$ where $\mathcal{R}$ is the set of available profiles.

Once the first resource $\overline{r}$ has to select its task, the profiles in $\mathscr{P}$ are scanned till 3 different pairs activity-identifier may be assigned to it. This allows the

system to provide the resource (at most) three different choices. Then $\overline{r}$ picks the case with identifier $\overline{i}$, and $\overline{r}$ executes the accordant activity $\overline{a}$. At that time, we remove all profiles in the set $\mathscr{P}$ in which the element $(\overline{i}, \overline{a}, \overline{r}, \overline{\Delta})$ is not present for some $\overline{\Delta}$. Then, the other resources can pick activities in order according to the retained profiles. This procedure, called **Exact Assignment (EA)**, provides a certain degree of freedom to the first resources, which can go quickly down as more and more resources pick cases for performance.

To overcome this problem, an alternative framework called **Approximate Assignment (AA)** is proposed: the only difference is that no profile is removed from $\mathscr{P}$. So the resulting Resource-task Assignment Set $\mathcal{S}$ may not coincide to those of any profile in $\mathscr{P}$. On the Approximate Assignment procedure, when the first resource $\overline{r}$ makes a choice of case with id $\overline{i}$ and activity $\overline{a}$, no profile is removed from $\mathscr{P}$. When the second resource makes a choice, (s)he presented the three best options in $\mathscr{P}$ without considering identifiers related to cases previously selected. Approximate assignment thus provides further freedom of choice, at the cost of potentially a lower global KPI improvement

5 Evaluation

The evaluation focuses on assessing the overall KPI improvements for two case studies (see Sect. 5.1). In particular, the comparison is done with respect to existing approaches, with specific emphasis on the work by Dumas et al. [14] where the framework exhibits a similar operational approach to that outlined within this documentation, although lacking the provision of multiple profiles, thereby terminating at the first greedy solution. We also focus on quantifying the degree of freedom, specifically assessing the extent to which resources possess the ability to choose their task, even if limited to a choice between two feasible options. This freedom's degree is a novel of our approach if compared with the state of the art. It indeed allows resources to pick among multiple alternatives.

Section 5.2 details the procedure for partitioning event logs into a training log and a test log. The training log is used to train the oracle function ψ, which plays a crucial role in our proposed methodology. Meanwhile, the test log is employed to evaluate the performance and effectiveness of our approach.

Subsequently, in Sect. 5.3, we discuss the assessment of recommendation quality and the level of resource autonomy achieved. Furthermore, Sect. 5.4 presents our evaluation method, which involves comparing the outcomes to a real-world scenario. Lastly, in Sect. 5.5, we analyze and interpret the results generated by our methodology.

5.1 Introduction to Use Cases

The validity of our approach was assessed using two different event logs with their associated use case. The first is so-called **Bank Account Closure** (BAC), a log referring to an Italian Bank Institution process that deals with the closures of bank accounts. From the bank's information system, we extracted an event

log containing 212,721 events containing 15 activities, 654 resources and 32,429 completed traces, divided into 14,593 for train and 17,836 for testing. For this log, we opted to consider the task in which the objective is to reduce the execution time of the instances, i.e. the KPI function $\mathcal{K}$ is equal to *Total Time* and the total number of generated profiles is 650,000.

The BPI challenge used the second log in 2013[4]. It is provided by Volvo Belgium and contains events from an incident and problem management system called **VINST**. We extracted 7,456 completed traces and 64,975 events. It contains 13 different activities that can be accomplished with 649 resources. In selecting traces from the log for training and testing, we get a training log of 3,355 traces and a test log of 4,101 traces.

For this case, we aim to avoid the occurrence of the activity *Wait-User*. The KPI value can be 1 or 0 if the activity occurs or not, while the Δ values related to the oracle function are evaluated as the difference by the probability of the activity occurring (i.e. $\Delta \in [0, 1]$). Note that one wants to reduce the activity-occurrence probability: the activity *Wait-User* is considered detrimental in terms of time and customer satisfaction. The total number of generated profiles is 140,000.

5.2 Train-Test Splitting Procedure

The starting point for an evaluation is an event log $\mathcal{L}$. In this section, to lighten the notation, we refer to $dom(\mathcal{L})$ as $\mathcal{L}$, and so referring to a log not as a function but as a set of trace identifiers in its domain. We first extract the training log $\mathcal{L}^{comp}$ for training the oracle function and, consequently, the recommender system. Then, we aim at creating the log $\mathcal{L}^{run}$ used for testing our system. To extract the training log $\mathcal{L}^{comp} \subset \mathcal{L}$ we compute the earliest time t_{split} such that 45% of the identifiers related to traces of $\mathcal{L}$ are completed. This allows us to define $\mathcal{L}^{comp}$ as the set of traces of $\mathcal{L}$ completed at time t_{split}, and consequently, define $\mathcal{L}^{run}$ as $\mathcal{L} \setminus \mathcal{L}^{comp}$. The traces of $\mathcal{L}^{run}$ are then truncated to a set $\mathcal{L}^{trunc}$ obtained from $\mathcal{L}^{run}$ by maintaining only a random percentage of events in each trace[5], this has been done for simulating running instances to which provide recommendations, using the set $\mathcal{L}^{run}$ for the evaluation of them.

5.3 Evaluation Metrics

The accuracy of recommending the resource r performs the activity a for the running case with identifier $i' \in dom(\mathcal{L}^{trunc})$ is evaluated as the average KPI of traces similar to it. Analytically, if $\mathcal{L}(i') = \sigma'$ and e such that $activity(e) = a$ and $resource(e) = r$:

$$score(\sigma', e) = avg_{\sigma \in Sim(\sigma', e, \mathcal{L}^{run})} \mathcal{K}(\sigma) \tag{1}$$

[4] https://www.win.tue.nl/bpi/doku.php?id=2013:challenge.

[5] The random percentage p was drawn from a uniform distribution $\mathcal{U}[25,\ 75]$, repeating the experiment for its stochastic validity.

where $Sim(\sigma', e, \mathcal{L}^{run})$ is the set of traces similar to $\sigma' \oplus \langle e \rangle$, namely

$$Sim(\langle e'_1, \ldots, e'_m \rangle, e, \mathcal{L}^{run}) = \{\sigma \in \text{cod}(\mathcal{L}^{run}) : \exists \sigma^p = \langle e_1, \ldots, e_{m+1} \rangle \in \text{prefix}(\sigma), \\ (activity(e_{m+1}), \text{resource}(e_{m+1})) = (\text{activity}(e), \text{resource}(e)), \\ \text{activity}(e_i) = \text{activity}(e'_i)\ \forall i \in \{1, \ldots, m\}\}$$

The score of the recommended action a to a resource r performing the running trace σ' is so the average KPI of traces similar to σ' for which the activity a has been performed by the resource r. This procedure is similar to the one used by de Leoni et al. in [2] and by Padella et al. in [7], adding the constraint about the recommended resource r to the similarity concept.

Typically, in the machine learning literature, the dimension of the train set is larger than the dimension of the test set. We chose this split ratio because using the accuracy evaluation proposed in Eq. 1, we evaluate a mean value on the output set of the function Sim that embodies the constraints related to the resource and the activity: this may lead to a small number of items on it, making the mean value evaluated statistically not significant.

As already mentioned, we also aim to give resources freedom in choosing which case and, consequently, activity to work on as next. Therefore, in our experiments, our goal is also to measure the resource freedom, hereafter defined as the number of resources that have given the freedom to choose the case to work on within a set that contains at least two cases. On this aim, we introduce the concept of **Freedom Score**, that is the ratio between the resources that had the possibility of choosing between at least two case-activity options in our assignment procedure (cf. Sect. 4.3) and the number of resources that can act on more than two running cases of $\mathcal{L}^{run}$. Analytically

$$Freedom\ Score(Assignment) = \frac{|\{r \in \mathcal{S} : r\ has\ chosen\ in\ Assignment\}|}{|\{r \in \mathcal{S} : r\ can\ act\ on\ more\ than\ one\ i \in\ dom(\mathcal{L}^{run})\}|} \quad (2)$$

The purpose of this function is to assess the degree of freedom of choice afforded to the resources by comparing it to the level of choice they typically have. A Freedom Score of 100% indicates that resources are granted complete freedom, while a score of 0% corresponds to no freedom.

5.4 Evaluation Methodology

The assessment of the system was carried out by trying to replicate actual organizational conditions. Therefore, we want to simulate how resources realistically interact with a recommender system.

1. Not all resource work at the same time, due to various factors such as shifts, vacations or other circumstances. Therefore, a Bernoulli distribution with a parameter of $p = 0.75$ is used to stochastically select a subset of resources: each individual element in the complete set of resources has a 75% probability of being designated as active and thus included in the subset.
2. Not all resources pick up a case to work on the same time, then we randomly shuffled the list of resources obtained at point 1, generating random arrival orders randomising the order in which resources pick their task.

Table 2. In the second column, the table presents the time values associated with the generation of the complete Profiles Ranking $\mathscr{P}$, representing approximately 10% of the total number of profiles that the framework can generate. The third column displays the absolute count of the generated profiles.

Case Study	Time needed	Total number of generated profiles
Total time on BAC	1 h and 40 min	650,000
Wait-User Occurrence on VINST	40 min	140,000

3. Resources are provided with a ranking of cases allowed to work on, ordered by expected KPI improvement. However, they do not necessarily pick the top element: research in Human-Computer Interaction has demonstrated a consistent pattern of user behavior when presented with a ranked list of options, as documented in [16]. In line with this study, we have adopted a stochastic resource selection behavior: Specifically, the probabilities of selecting the first, second, and third options are 61%, 24%, and 15%, respectively.

Since the points 1–3 in the list above rely on sampling from distributions (e.g. the Bernoulli distribution at point 1), the procedure has been repeated: we extracted 10 values from the Bernoulli distribution described at point 1 and, for each of these values, the random shuffling has been done 10 times. It follows that, in total, we repeated the evaluation 100 times.

The improvements by our framework has been evaluated by applying the formula in Eq. 1 to the recommendations provided to the traces relatives to the identifiers in $\mathcal{L}^{trunc}$ using the two assignment procedures defined in Sect. 4.3 and then comparing this scores with the real process executions from $\mathcal{L}^{run}$.

5.5 Results Analysis

For each of the 10 subsets of existing resources obtained at point 1 of the evaluation methodology (cf. Sect. 5.4), we computed the total number of potential profiles. However, the time needed to compute them all is practically not feasible and hence we use our framework to only generate up to 10% of them in experiments, with increments of 1%.

Due to differences in the number of activities, resources, and cases in the logs, the computational times varied. All the generations were executed on a workstation equipped with a 16-core AMD Ryzen 7 4700G processor unit and 16 GB RAM, which were divided into 12 different threads. Table 2 shows for each case study, the time to generate this 10% of profiles. This threshold represents a justifiable value since the tree procedure described in the Sect. 4.2 follows a greedy approach: on it, profiles are initially generated in a stochastic manner and subsequently filtered to eliminate duplicates. As the number of generated profiles augments, the likelihood of encountering new profiles decreases, leading to an exponential rise in the time required for generating new profiles.

The experiments' results in terms of KPI values are shown in Fig. 4, which illustrates how the improvement is linked to the percentage of profiles that are

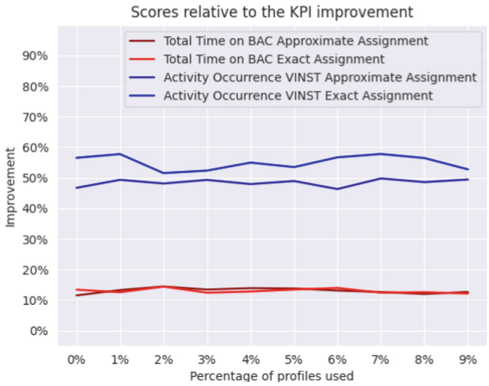

Fig. 4. Results related to the KPI improvement for both case studies and the two assignment techniques. On the x-axis there is the percentage of the profile used for running the two algorithms, while in the y-axis the average KPI improvement on the whole $\mathcal{L}^{run}$ evaluated as defined in Eq. 2 is shown.

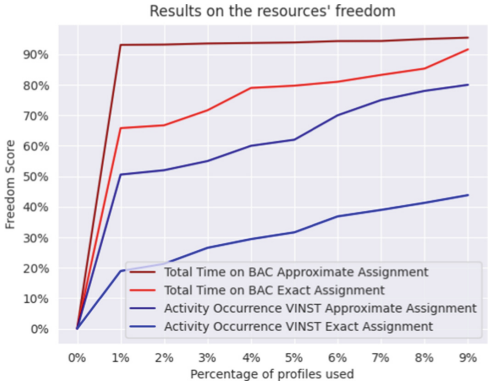

Fig. 5. Results related to the freedom left to the resources for both case studies and the two assignment techniques. On the x-axis there is the percentage of the profile used for running the two algorithms, while on the y-axis the Freedom Score as defined in Eq. 2 is shown.

generated, for the BAC and VINST case studies, and using our two approaches to compute the set of profiles. The results show that it is sufficient to generate few profiles to obtain significant KPI improvements. For the case study of reducing total execution time on BAC, there is no significant variation in outcomes between the Exact and Approximate assignment techniques. In the conducted case study focusing on optimizing process performance in the BAC system, we achieved an improvement of 58%. This improvement translates to a reduction in the total execution time of all active traces from 179, 430 h to 76,873 h. We successfully minimized the overall processing time by implementing the proposed

measures, leading to significant efficiency gains. Furthermore, our investigation targeted the reduction of the *Wait-User* activity occurrences within the VINST dataset. The initial analysis identified 631 traces in which this activity took place. Through the implementation of optimized strategies, the occurrence of *Wait-User* decreased to 486 traces. This reduction highlights the effectiveness of the proposed approach in streamlining the process and minimizing potential bottlenecks associated with this specific activity of 12%.

A larger number of profile generation may still remain relevant to allow resources a larger degree of freedom to choose the case, and hence the intervention, to work on. Figure 5 shows how the Freedom Score increases with larger number of profiles that are generated. The algorithm for approximate assignments seem to consistently allow larger freedom. It even achieves a Freedom Score of 90% after generating only 1% of the profiles for the BAC case study.

The Approximate Assignment method was indeed designed to provide resources with more freedom of choice, a goal successfully achieved in both case studies. The approximate assignment was also able to achieve the same amount of KPI improvement as the exact assignment: therefore, the Approximate-Assignment algorithm is certainly preferable for the BAC case study. For the VINST case study, the Exact-Assignment method provides results that are around 10% better than the Approximate-Assignment method, which suggests the Exact Assignment method is preferable.

It follows that opting for the Exact-Assignment or for the Approximate-Assignment method may depend on the case study. Therefore, the choice requires to conduct a prior assessment based on training and testing phases, as conducted in the experiments discussed in this paper.

Last but not least, *we aim to compare our results with those obtained by the latest advantages in prescriptive process analytics, and we have carried on a comparison with respect to the approach by Dumas et al.* [14]. *The approach by Dumas et al. corresponds to the scenario in which only the first profile is generated, analogously to what discussed in Sect. 4.1. The conducted experiments have shown that the creation of multiple profiles and their ranking provides a further improvement by 6.2% and 1.9% for the VINST and BAC case studies, respectively.* It is worthwhile noting here that Dumas et al. use a different prescriptive-analytics oracle function. However, a fair comparison requires to use the same oracle function, to put aside any difference due to the choice of the oracle function. This motivates why we use our oracle function in both of scenarios, namely only using the first profile, or conversely leveraging on the profile ranking.

6 Conclusions

Process-aware Recommendation systems are a class of information systems that provide support to process stakeholders to achieve better results for the running cases. The module that suggests effective interventions is obviously the core module in this class of systems. The intervention for a running case typically

consists in suggesting a certain activity to be performed as next, as well as the resource to which this activity should be given for performance. Existing techniques propose interventions to single running cases in isolation, making the choice of interventions local to single cases. However, resources are shared among cases, and hence an allocation of resources and interventions should be deal with as a global optimization problem, where all cases requiring interventions are considered altogether.

This paper has put forward a framework that tackles the global optimization problem. It is clear that the complexity of the problem is NP-hard, and hence finding an optimal solution is intractable when hundreds of cases are running at the same time, and also hundreds of resources are involved. We thus propose two approximated algorithms that aim to find sub-optimal solutions. The algorithm returns a number of alternative user profiles, each of which consists in a set of assignments of activities to resources, with the constraint that a resource can only work on with a case within a profile. These profiles are then ranked with the expected outcome improvement, measured in terms of KPIs. Each resource is then offered the interventions ordered by descending ranking of the profile of which those interventions are part.

Among the advantages of our proposal, it is worthwhile mentioning that, while most of existing approaches impose an assignment of cases and activities to resources, we provide process actors with a certain degree of freedom in choosing what to work on. This freedom is clearly very beneficial in the context of recommender systems: vice versa, imposing an assignment may potentially incur in the risk of having resources to act on cases independently and regardless of the recommendations.

Experiments were conducted with two real-life case studies, emulating how humans would pick offered interventions in an order list. This emulation was based on behavioral models described in the human-computer interaction literature [16]. The results illustrate a significant improvement with respect to frameworks that aim to improve the outcome of running cases in isolation. So did we compare with the approach by Dumas et al. [14], which is the only approach that we found that is able to provide a global KPI improvement: our framework provides a further improvement by 6.2 and 1.9% for the two case studies.

Acknowledgement. The PhD. scholarship of Mr. Padella is partly funded by IBM Italy, and by the BMCS Doctoral Program, University of Padua. This research is also supported by the Department of Mathematics, University of Padua, through the BIRD project "Data-driven Business Process Improvement" (code `BIRD215924/21`).

References

1. Comuzzi, M.: Ant-colony optimisation for path recommendation in business process execution. J. Data Semantics **8**(2), 113–128 (2019)
2. de Leoni, M., Dees, M., Reulink, L.: Design and evaluation of a process-aware recommender system based on prescriptive analytics. In: 2020 2nd International Conference on Process Mining (ICPM) (2020)
3. Weinzierl, S., Dunzer, S., Zilker, S., Matzner, M.: Prescriptive business process monitoring for recommending next best actions. In: Business Process Management Forum (2020)
4. Metzger, A., Kley, T., Palm, A.: Triggering proactive business process adaptations via online reinforcement learning. In: Fahland, D., Ghidini, C., Becker, J., Dumas, M. (eds.) BPM 2020. LNCS, vol. 12168, pp. 273–290. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58666-9_16
5. 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. Knowl. Inf. Syst. **64**, 02 (2022)
6. Bozorgi, Z.D., Teinemaa, I., Dumas, M., Rosa, M.L., Polyvyanyy, A.: Prescriptive process monitoring for cost-aware cycle time reduction. In: 2021 3rd International Conference on Process Mining (ICPM) (2021)
7. Padella, A., de Leoni, M., Dogan, O., Galanti, R.: Explainable process prescriptive analytics. In: 2022 4th International Conference on Process Mining (ICPM), pp. 16–23 (2022)
8. Shapley, L.S.: A value for n-person games. RAND Corporation, no. 28 (1953)
9. Cabanillas, C., Schönig, S., Sturm, C., Mendling, J.: Mining expressive and executable resource-aware imperative process models. In: Gulden, J., Reinhartz-Berger, I., Schmidt, R., Guerreiro, S., Guédria, W., Bera, P. (eds.) BPMDS/EMMSAD -2018. LNBIP, vol. 318, pp. 3–18. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91704-7_1
10. Havur, G., Cabanillas, C.: History-aware dynamic process fragmentation for risk-aware resource allocation. In: Panetto, H., Debruyne, C., Hepp, M., Lewis, D., Ardagna, C.A., Meersman, R. (eds.) OTM 2019. LNCS, vol. 11877, pp. 533–551. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-33246-4_33
11. Zhao, W., Yang, L., Liu, H., Wu, R.: The optimization of resource allocation based on process mining. In: Huang, D.-S., Han, K. (eds.) ICIC 2015. LNCS (LNAI), vol. 9227, pp. 341–353. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-22053-6_38
12. Huang, Z., van der Aalst, W., Lu, X., Duan, H.: Reinforcement learning based resource allocation in business process management. Data Knowl. Eng. **70**(1), 127–145 (2011). https://www.sciencedirect.com/science/article/pii/S0169023X1000114X
13. Park, G., Song, M.: Prediction-based resource allocation using LSTM and minimum cost and maximum flow algorithm. In: International Conference on Process Mining (ICPM) 2019, pp. 121–128 (2019)
14. Shoush, M., Dumas, M.: When to intervene? prescriptive process monitoring under uncertainty and resource constraints. In: Di Ciccio, C., Dijkman, R., del Río Ortega, A., Rinderle-Ma, S. (eds.) Business Process Management Forum, pp. 207–223 Springer, Cham (2022). https://doi.org/10.1007/978-3-031-16171-1_13

15. de Leoni, M.: Foundations of Process Enhancement, pp. 243–273. Springer, Cham (2022)
16. Joachims, T., Granka, L., Pan, B., Hembrooke, H., Gay, G.: Accurately interpreting clickthrough data as implicit feedback. In: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ser. SIGIR 2005, pp. 154–161. Association for Computing Machinery, New York (2005)