Experience-Based Resource Allocation for Remaining Time Optimization

Alessandro Padella[1], Felix Mannhardt[2], Francesco Vinci[1], Massimiliano de Leoni[1], and Irene Vanderfeesten[3]

University of Padua, Padua, Italy
alessandro.padella@phd.unipd.it
deleoni@math.unipd.it
francesco.vinci.1@phd.unipd.it
Eindhoven University of Technology, Eindhoven, The Netherlands
f.mannhardt@tue.nl
KU Leuven, 3000 Leuven, Belgium
irene.vanderfeesten@kuleuven.be

Abstract. Prescriptive process analytics aims to suggest interventions for those process instances that are predicted to not achieve a satisfactory outcome. Typical interventions are recommending a task to be performed by a specific resource. State-of-the-art prescriptive resource allocation techniques typically propose interventions that allocate the best-fitting resources at a given time. This may result in those resources to become more skilled at the task over time whereas other less experienced resource are rarely allocated. In the long run, such system may result in a unbalanced situation in which some expert resources are overloaded with very high workload and the less experienced resource are assigned fewer tasks and fail to improve. This paper proposes an approach for resource allocation to process instances that aims at a more balanced workload distribution among the resources, even if this means slightly lower process improvements in the short term. Experiments on event logs related to two real processes show that we indeed achieve a more balanced workload distribution, which often yields an overall higher improvement of the whole set of running process instances.

Keywords: Process Prescriptive Analytics · Resource Allocation · Recommender Systems · Machine Learning · Workload Distribution

1 Introduction

Process mining aims to discover and improve processes, on the basis of the analysis of a process' transactional data that record how single executions have been carried out. Within the realm of process mining, prescriptive process analytics focuses on running process executions and tries to recover the "risky" executions that, otherwise, are predicted to achieve unsatisfactory outcomes (high costs, poor customer satisfactions, etc.). In prescriptive process analytics, the system advises interventions at run-time, typically in the form of suggesting given resources to perform certain recovery activities.

Prescriptive process analytics is currently gaining momentum, and several approaches have been proposed over the years (cf. survey by Kubrak et al. [18]). While a large body

of research has mainly focused on suggesting the next activity to work on, some works also consider the resource availability and optimize the resource assignment to these recovery activities (see Sect. 2).

In this paper, we want to move forward from the literature that just focuses on allocating the best-fitting resources to activities for selected running process executions. In many situations, there is a clear correlation between the performances of activities by resources and the experience of the resources themselves. When recommending based on the historical data on resource performance, this yields to a vicious cycle in which the best-fitting resources are always assigned activities, and they thus become even more experienced and better performing. A harmful consequence of such recommendations may be that the less experienced resources are never given the chance to gain experience and become better at performing recovery activities: they are always being kept aside. Note how this uneven distribution of activity work is intrinsically unfair because some resources are largely idle, while others are heavily overloaded, increasing the queue of the activities that they have to perform, thus ultimately negatively affecting the overall process performances. Considering such fairness problems in performance analysis has been advocated previously [1].

The paper aims to introduce an approach that balances the necessity for process executions to be improved at their best, on the one hand, and the long-term goal to enlarge the repertoire of experienced resources, on the other hand. The BPR best practices discussed in [27, 9] already advised to "consider to broaden the skills of resources" (i.e. generalist vs specialist profiles) to improve flexibility during process execution and to have a better utilization of resources. Pika et al. suggested that analysis of resource behaviours may provide insights for effective human resource development strategies, and resource and process performance improvement [24]. Vanderfeesten and Grefen also acknowledge the importance of periodically allocating difficult tasks to train resources and enlarge their experience with different tasks [29]. This long-term goal is overlooked by the predictive-analytics literature, briefly discussed above and in Sect. 2, but it can potentially bring clear benefits for the organization to which the process belongs.

Currently, our approach described in Sect. 4 focuses on prescribing which activities to perform on risky cases and to which resources to assign them, with the aim to reduce the remaining time to the conclusion of process executions. However, the approach can be easily extended to other KPIs. Specifically, the approach leverages on building a machine-learning model that estimates the remaining time together with considering the learning curve of resources. This model is then used to recommend the process executions on which to work as next, the activities to carry out for them, and the set of resources to whom to allocate these activities.

Experiments reported in Sect. 5 were conducted for two real-life processes, for which transactional data with suitable resource information was available. One focuses on an Italian banking institution, and another refers to a Dutch financial institution. For the latter, a concept-drift was present: therefore, we conducted experiments with the portion of transition data before and after the drift, leading to two alternative processes. The results show that our approach indeed leads to a better workload distribution, which typically translate to reducing the remaining time to complete process executions. The

latter illustrates how allowing resources to gain experience may provide a long-term general benefit for the organization in which the process is being executed.

2 Related Works

Advanced dynamic resource allocation in business process management has recently gained increased attention. Several authors have identified additional information about the (human) resources that may be employed to prioritize specific resources and to enhance allocation decisions [29, 3, 26, 7]. Additional information on a worker's expertise [4], preferences [5, 31], and abilities [10] can be incorporated in allocation decisions.

Also, experience is mentioned as one of the additional factors to take into account [3, 26, 11]. Experience indicates the working time since a resource joined the enterprise, reflecting the familiarity of resources with the enterprise [30]. It is often measured in number of days or years on the job [7, 24, 31], although [16, 15] propose that experience may be measured through (i) resumé-like metrics such as 'number of years on the job', 'total time employed', 'number of jobs/functions', 'number of organizations', and (ii) metrics based on historical task execution data from process-aware information systems such as 'number of times performing a task', 'task success/failure rate', and 'task variety'. Also [11] define experience as 'the number of times an employee has executed a work item, been involved in a case, and interacted with others', and Kim et al [17] considered for instance 'the number of work items', 'the number of cases for which at least one work item executed' and 'the number of unique tasks executed'. In this paper we adopt this second perspective focusing on historical task execution data and analyse the development of a worker's experience in terms of time needed to execute the task.

Some of the existing works on advanced dynamic resource allocation propose to collect this additional information based on historical process execution data (i.e., event logs) [4, 31]. Ly et al. focus on staff assignment rules [21], Huang et al. mine association rules, and resource behavior (preference, availability, competence and cooperation) [12, 13], Kumar et al. discover collaboration patterns [19], Pika et al. propose to mine actor profiles including expertise and experience [24], and Kim et al. provide a framework for extracting features from event logs capturing resource experiences, such as frequency, performance and busyness, for predictive process monitoring [17]. In this paper, we also use historical data but derive a more complex understanding of experience.

There are many approaches to optimize resource allocation decisions using various computation and artificial intelligence techniques, ranging from heuristics, to linear programming and genetic algorithms [26, 14]. In some of these algorithms, a worker's experience (or sometimes called past performance) is used to make allocation recommendations [4, 31, 17]. However, most of these approaches focus on a local optimization decision, i.e. finding the best matching resource for a specific task without considering the other running process instances that also need to be optimized [14]. The works reported in [28, 22] try to go beyond and globally optimize the resource allocation for all process instances that require recovery interventions. They still aim to optimize the outcome of the running process instances, overlooking the problems related to *(i)* a fair work distribution and *(ii)* facilitating the acquisition of the experience by the resources. The solution presented in this paper addresses these gaps.

3 Preliminaries

The starting point for amy process mining technique is an *event log*. An event log is a multiset of *traces*. A 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 by certain *resources* at a specific *timestamps* together with the process *attributes* manipulated.

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

A trace is a sequence of events. When attributes are given the same assignment the same event can occur in different traces. Since full traces can also appear multiple time, we define an event log with a function that assigns unique trace identifiers to traces.

Definition 2 (Traces & Event Logs). *Let $\mathcal{E} = \mathcal{A} \times \mathcal{T} \times \mathcal{R} \times (\mathcal{V} \not\to \overline{\mathcal{W}})$ be the universe of events. Let $\mathcal{I}$ be the universe of the case identifiers. Let a trace σ be 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 id of a log trace, returns the sequence of events related to the process instance with the identifier id, i.e. $\mathcal{L} : \mathcal{I} \to \mathcal{E}^*$.*

Given an event $e = (a, t, r, v)$, we use the following shortcuts: $act(e) = a$, $time(e) = t$, $res(e) = r$ and $var(e) = v$. Given a trace $\sigma = \langle e_1, \ldots, e_n \rangle$, $prefix(\sigma)$ denotes the set of all prefixes of σ, including σ: $prefix(\sigma) = \{\langle\rangle, \langle e_1 \rangle, \langle e_1, e_2 \rangle, \ldots, \langle e_1, \ldots, e_n \rangle\}$.

In developing our technique, it is imperative to delineate the goal of our recommendation. We focus on minimizing the remaining time of a trace. It can be defined as a function over a given trace $\langle e_1, \ldots, e_n \rangle = \sigma \in \mathcal{E}^*$ and a timestamp $t \in \mathbb{N}$ that returns the difference between the last timestamp related to σ, and the given timestamp t. We use the notation $\mathcal{R}_{time}(\sigma, t) = time(e_n) - t$ in the remainder. Note that the remaining time is assumed to be computed a posteriori when the execution is completed and leaves a complete trail of the timestamps for a certain trace σ.

Hereafter, we assume that the technique provides recommendations in a fixed timestamp t_{sys} for traces σ' that are not considered as completed at that timestamp, i.e. running traces. With an abuse of notation, we indicate $\mathcal{R}_{time}(\sigma', t_{sys}) = \mathcal{R}_{time}(\sigma')$, namely the remaining time value for the running trace σ', given the timestamp t_{sys} in which the recommendation is delivered. This is to simplify notation by assuming that the recommendations are provided for the (running) traces of a log as in a real-life deployment in an organisation.

To understand which cases are risky and need recommendations, we first want to understand which running process instances are leading to a high remaining time. Towards this goal, we define the **Ranking function** that takes as input an event log $\mathcal{L} : \mathcal{I} \to \mathcal{E}^*$,

[1] A timestamp could, e.g., be represented as UNIX epoch.

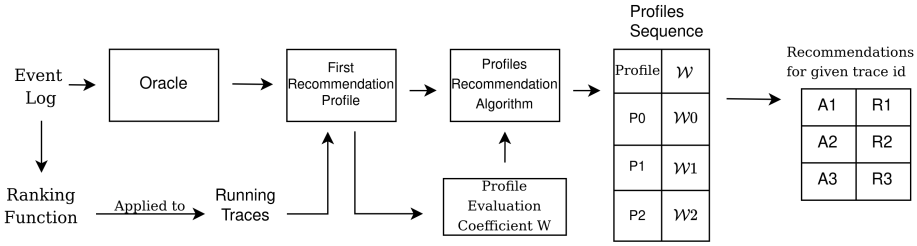


Fig. 1: Overview of the proposed approach.

consisting of running traces, and generates an output sequence of trace indices, i.e., a ranking, in descending order based on the remaining execution time.

Definition 3 (Ranking function). *Let $\mathcal{I}$ be the universe of case identifiers. Let $\mathcal{E}^*$ the possible traces and $\mathcal{L} : \mathcal{I} \to \mathcal{E}^*$ an associated event log. Let $\mathcal{R}_{time} : \mathcal{E}^* \times \mathbb{N} \to \mathbb{N}$ the corresponding remaining time function. The Ranking function $Rank : F(\mathcal{I}, \mathcal{E}^*) \to (\mathcal{I} \times \mathbb{N})^*$, given an event log with $|\mathcal{I}| = n$ identifiers, returns a list of pairs*

$$\{(id_1, \mathcal{R}_{time}(\mathcal{L}(id_1))), \ldots, (id_n, \mathcal{R}_{time}(\mathcal{L}(id_n)))\}$$

sorted decreasing by $\mathcal{R}_{time}$.[2]

To lighten the notation, we refer to $(id_i, \mathcal{R}_{time}(\mathcal{L}(id_i))$ as $(id_i, \mathcal{R}_{time}(id_i))$. Indeed, we can generate such ranking by applying the remaining time function to every, still running, trace of the event log.

4 An Approach for Experience Based Resource Allocation

We introduce a technique to recommend resources which activity to perform next and on which running trace while taking into account their experience and the overall workload distribution. As motivated earlier, always allocating the most experienced resource for each process instance without considering the workload distribution and the changing experience of resources in performing tasks leads to more frequently allocating work to already experienced resources. Such uneven and unfair distribution of work has two harmful consequences. In the short term, the most experienced resources are overloaded, increasing the queue of tasks they have to perform and leaving other resources idle. In the long run, it may induce a vicious circle: already experienced resources will further improve their skills and be faster in performing tasks but risk being overloadedMeanwhile, the less utilized resources fail to gain experience and will not be recommended by the system in the future since they are considered not competitive.

Our work's principal aim is to enhance the efficiency of overall system efficiency by proposing suitable activities and resources to reduce the aggregate remaining time of the (running) traces. Figure 1 depicts our approach. First, we get a ranking of traces based

[2] $F(\mathcal{I}, \mathcal{E}^*)$ is the space of the functions from the set of identifiers $\mathcal{I}$ to the set of traces $\mathcal{E}^*$.

on the Ranking Function obtained from an event log. Then, using an Oracle function learned from the resource experience encoded in the event log, we derive an initial Recommendation Profile taking into account the ranking. This provides a possible activity and resource allocation for each running trace. To balance the allocation, we define a time-workload coefficient and use it to generate a sequence of increasingly better profiles respecting all the desired aspects. Finally, a recommendation based on the best generated Recommendation Profiles is provided.

To realize such a system, we first define the **Task Duration function**, which predicts the time a given resource needs to accomplish a certain task in a given process state.

Definition 4 (Task Duration Function). *Let $\mathcal{A}$ be the set of process activities. Let $\mathcal{R}$ be the set of the possible resources. Let $\mathcal{L} : \mathcal{I} \to \mathcal{E}^*$ be an event log defined over a set $\mathcal{I}$ of identifiers. The Task Duration function $\lambda : \mathcal{I} \times \mathcal{A} \times \mathcal{R} \times \mathbb{N} \to \mathbb{N}$ returns a value $\lambda(id, a, r, t)$, representing the amount of time units needed by the resource r for executing the activity a as next, for the trace $\mathcal{L}(id) = \langle e_1, \ldots, e_i \rangle$, starting at the timestamp $time(e_i) + t$.*

We omit a, id, and r from $\lambda(id, a, r, t)$ and use λ_t when they are clear from context.

Given a suitable Task Duration function λ, we can vary the input value t to see how the output values of the function change based on the experience gained throughout the whole process. Using this function as a starting point, it is possible to define the **Experience Based Oracle** on which the system is based. This function returns the remaining time for a running trace given the next activities and resources along with the expected execution times of the resources for the associated task at different future time units. In fact, varying the input t of the function λ, in a set of p future timestamps, provides an overview of the different execution times in the different timestamps. This results in a curve representing how the given resource will improve/worsen its efficiency during the time, resulting in mapping the experience that they gain.

Definition 5 (Experience Based Oracle). *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, $\mathcal{R}$ the set of the available resources. Let $\mathcal{I}$ be the set of trace identifiers. Let $\mathcal{R}_{time} : \mathcal{E}^* \times \mathbb{N} \to \mathbb{N}$ the remaining time function, and $\lambda : \mathcal{I} \times \mathcal{A} \times \mathcal{R} \times \mathbb{N} \to \mathbb{N}$ the Task Duration function. The Experience Based Oracle is a function $\psi : \mathcal{I} \to 2^{(\mathcal{A} \times \mathcal{R} \times \mathbb{N} \times \mathbb{N}^p)}$ such that $\psi(id)$ returns $\{(a_1, r_1, r_{time\_1}, \vec{\lambda}_1), \ldots, (a_m, r_m, r_{time\_m}, \vec{\lambda}_m)\}$ with $m \leq |\mathcal{R}|$, indicating that activity a_i is recommended and, if performed by r_i, will lead to a total remaining time $r_{time\_i} = \mathcal{R}_{time}(id_i)$ for the input trace, and the resource r_i will take $\vec{\lambda}_i = (\lambda(id, a_i, r_i, t_{i1}), \ldots, \lambda(id, a_i, r_i, t_{ip}))$ time units for performing the activity a_i starting in p different timestamps. Also, $\forall i, j \in \{1, \ldots, m\}$, $r_i = r_j \iff i = j$, meaning that a resource can only be recommended once.*

Table 1 provides an illustrative example of the output generated by the Experience Based Oracle ψ on a running trace. As shown, if the organization were to follow the recommendation in the second row, which allocates resource `User_56` to perform `O_Returned`, then the remaining time is 45 days. Additionally, considering the experience of the resource the projected execution times for the allocated task is 5 days

Table 1: Example of output of the Experience-Based Oracle (ψ) for a specific trace identifier. In the first 3 columns, activities, resource and remaining times are shown. The rightmost three columns indicate the execution times for the given resources to perform the associated activity if starting it in 24, 120 and 480 hours, respectively.

Activity	Resource	$\mathcal{R}_{time}$	λ_{24}	λ_{120}	λ_{480}
O_Returned	User_119	38d	2d 3h	2d	2d
O_Returned	User_56	45d	5d 2h	5d 2h	3d 5h
A_Submitted	User_1	46d	5d 6h	4d 4h	4d 2h
O_Create Offer	User_13	48d	12d 2h	12d 2h	10d 5h

and 2 hours if the recommendation is initiated within 24 or 120 hours after the time t_{sys} in which the system is and 3 days and 5 hours if commenced after 480 hours after it.

Such Experience Based Oracle Function ψ has been implemented in several ways, (cf. Section 2), but none of the algorithms has integrated it with the Task Duration Function. We opted to use the prescriptive-analytics proposal discussed in [23], by expanding it to incorporate the λ function. This approach employs a Transition System to propose alternatives for the subsequent activity recommendation, ensuring that the suggested activities are coherent and feasible.

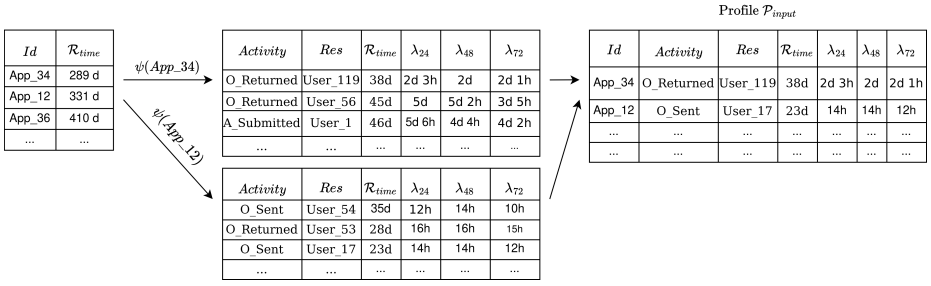
The Experience Based Oracle function provides an overview of the possible recommendations for a given trace. However, our approach aims to provide recommendations that consider both the state of the system and how it will evolve, and not the single distinct traces. To take into account all of these aspects, the Profile concept in [22] has been extended, incorporating the execution time values from the function λ and generating it with a different algorithm. This results in the definition of **Recommendation Profile**.

Definition 6 (Recommendation Profile). *Let $\mathcal{I}$ the set of trace identifiers, $\mathcal{A}$ the set of activities, $\mathcal{R}$ the set of resources, $\mathcal{R}_{time}$ the remaining time function and $\lambda : \mathcal{I} \times \mathcal{A} \times \mathcal{R} \times \mathbb{N} \to \mathbb{N}$ the Task Duration function. A Recommendation Profile is a set $\mathcal{P} \subset (\mathcal{I} \times \mathcal{A} \times \mathcal{R} \times \mathbb{N} \times \mathbb{N}^p)$ defined as a set of tuples $(id, a, r, r_{time}, \vec{\lambda})$, in which every running instance corresponding to the case identifier id is recommended to be assigned to a resource r and to perform the activity a, leading to a remaining time of r_{time}. Furthermore, the recommendations are associated with p execution times if starting in different timestamps after the last event, represented by the values in $\vec{\lambda} \in \mathbb{N}^p$.*

The definition of a Recommendation Profile encompasses all the aspects addressed in this paper. In fact, the first part ef each vector (id, a, r, r_{time}) includes recommendations associated with corresponding remaining times if the recommendation is followed, while the last p elements of each tuple report the expected time for the resources to complete the corresponding task starting in different time units after the last event occurred.

4.1 Profiles Generation

The generation of a Recommendation Profile $\mathcal{P}_{input}$ starts with an event log $\mathcal{L} : \mathcal{I} \to \mathcal{E}^*$ defined on a set of n trace identifiers in $\mathcal{I}$. Initially, the Ranking Function $Rank$:



Id	$\mathcal{R}_{time}$
App_34	289 d
App_12	331 d
App_36	410 d
...	...

$Activity$	Res	$\mathcal{R}_{time}$	λ_{24}	λ_{48}	λ_{72}
O_Returned	User_119	38d	2d 3h	2d	2d 1h
O_Returned	User_56	45d	5d	5d 2h	3d 5h
A_Submitted	User_1	46d	5d 6h	4d 4h	4d 2h
...	...	...	...	...	...

$Activity$	Res	$\mathcal{R}_{time}$	λ_{24}	λ_{48}	λ_{72}
O_Sent	User_54	35d	12h	14h	10h
O_Returned	User_53	28d	16h	16h	15h
O_Sent	User_17	23d	14h	14h	12h
...	...	...	...	...	...

Id	$Activity$	Res	$\mathcal{R}_{time}$	λ_{24}	λ_{48}	λ_{72}
App_34	O_Returned	User_119	38d	2d 3h	2d	2d 1h
App_12	O_Sent	User_17	23d	14h	14h	12h
...	...	...	...	...	...	...
...	...	...	...	...	...	...

Fig. 2: Profile generation procedure. On the top left corner, a tabular example of the output of $Rank(\mathcal{L})$. Then, for the first two trace identifiers, ψ is evaluated. From the respective outputs, the element with the lowest $\mathcal{R}_{time}$ is added to the Profile $\mathcal{P}_{input}$.

$F(\mathcal{I}, \mathcal{E}^*) \rightarrow \mathcal{I} \times \mathcal{E}^*$, defined in Section 3, is first applied to the log, obtaining the output $\{(id_1, \mathcal{R}_{time\_1}), \ldots, (id_n, \mathcal{R}_{time\_n})\}$, then, for every element $(id_i, \mathcal{R}_{time\_i}) \in Rank(\mathcal{L})$, 3 steps are iterated:

1. The Experience Based Oracle ψ is applied, resulting in
$$\psi(id_i) = \{(a_{i1}, r_{i1}, \mathcal{R}_{time\_i1}, \vec{\lambda}_{i1}), \ldots, (a_{im}, r_{im}, \mathcal{R}_{time\_im}, \vec{\lambda}_{im})\}.$$
2. From the output of the ψ, the $\hat{k} - th$ element $(a_{i\hat{k}}, r_{i\hat{k}}, \mathcal{R}_{time\_i\hat{k}}, \vec{\lambda}_{i\hat{k}})$ with $\hat{k}$ such that $\mathcal{R}_{time\_i\hat{k}} = min_{k=1,\ldots,m}(\{\mathcal{R}_{time_{ik}}\})$ is picked.
3. Then, the element $(id_i, a_{i\hat{k}}, r_{i\hat{k}}, \mathcal{R}_{time\_i\hat{k}}, \vec{\lambda}_{i\hat{k}})$ is added to the profile $\mathcal{P}_{input}$.

This procedure allows the generation of a Recommendation Profile that optimises only the remaining time, not considering other aspects like the workload distribution or the experience that resources gain. The example in Figure 2 depicts the creation of a Recommendation Profile $\mathcal{P}_{input}$. On the top left, the output of the Ranking Function $Rank$ is depicted: it associates the first 3 running traces identifiers App_34, App_12 and App_36 to their respective remaining time values $289d$, $331d$ and $410d$, sorted descending. Then, for the first trace identifier App_34, the function $\psi(App\_34)$ is evaluated, resulting in output with multiple resources and activities, with the vector $\vec{\lambda} = (\lambda_{24}, \lambda_{48}, \lambda_{72})$ containing the execution times if executing the activity in 24, 48 or 72 hours. From this, the tuple $(O\_Returned, User\_119, 38d, 2d\ 3h, 2d, 2d\ 1h)$ is chosen because it is the one that leads to the lowest value of $\mathcal{R}_{time}$. Finally, the tuple $(App\_34, O\_Returned, User\_119, 38d, 2d3h, 2d, 2d\ 1h)$ is added to the Profile. The same procedure is iterated with the other trace identifier values of $Rank(\mathcal{L})$.

This method associates the most experienced resources with their best activity, with running traces leading to the lowest remaining times. The first profile starts with the values obtained from the Ranking function's output. However, it only aims to optimise the remaining time without considering the distribution of work among the resources. To enhance this allocation with considering the workload distribution, we define the **Time-Workload Coefficient** $\mathcal{W}$ of a profile:

$$\mathcal{W}(\mathcal{P}) = C_1 * \underset{i=1,\ldots,n}{mean}\left(\mathcal{R}_{time\_i}\right) + C_2 * \underset{j=1,\ldots,p}{mean}\left(\underset{i=1,\ldots,n}{std}\left(\vec{\lambda_{ij}}\right)\right) + C_3 * \left(1 - \frac{\#res}{|\mathcal{R}|}\right)$$

This coefficient allows us to evaluate the quality of a Recommendation Profile under both short-term efficiency and workload perspectives. The coefficient assessing a profile is partitioned into three terms each weighted by a corresponding C_i coefficient to facilitate an equal contribution of all three components. The first term computes the mean of the remaining times $\mathcal{R}_{time\_i}$ in the Recommendation Profile. The second term considers all the standard deviations given an expected time j and takes the average of them. This value represents the average standard deviation of the execution times of the resources. The third term denotes the count of inactive resources within the profile.

We use the standard deviation here since it indicates how much the values of the vector spread from the mean. In our case, low values represent that the time required for resources to complete the task is consistent and not widely dispersed from the average time. The count of inactive resources is considered to prevent scenarios where only a few resources are recommended. It is noteworthy that a lower Time-Workload coefficient value indicates a more desirable Recommendation Profile. Specifically, our objective is to minimize the remaining time, maintaining a low standard deviation in resource workload distribution while having a limited number of inactive resources.

4.2 Generation of Profiles Sequence

Starting from the single generated Recommendation Profile, we aim to generate a sequence $\mathcal{S}$ of optimal Recommendation Profiles with regard to the just defined time-workload coefficient. Then, the recommendation for resource-task allocation is provided from these optimal profiles with a minimal time-workload coefficient.

Given a starting profile $\mathcal{P}_{input}$, and a sequence of Recommendation Profiles initialized as $\mathcal{S} = \{(\mathcal{P}_{input}, \mathcal{W}(\mathcal{P}_{input}))\}$, we use the following procedure to generate multiple optimal recommendation profiles:

1. For the profile $\mathcal{P}_{input}$ the Time-Workload coefficient $\mathcal{W}(\mathcal{P}_{input})$ is evaluated.
2. A random number $\overline{k} \in \{1, \ldots, n\}$ is extracted.
3. For the $\overline{k} - th$ element $(id_{\overline{k}}, a_{\overline{k}}, r_{\overline{k}}, \mathcal{R}_{time\_\overline{k}}, \vec{\lambda}_{\overline{k}})$ of the profile $\mathcal{P}_{input}$, the Experience Based Oracle $\psi(id_{\overline{k}})$ = is evaluated.
4. From the output

$$\psi(id_{\overline{k}}) = \{(a_{\overline{k}1}, r_{\overline{k}1}, \mathcal{R}_{time\_\overline{k}1}, \vec{\lambda}_{\overline{k}1}), \ldots, (a_{\overline{k}m}, r_{\overline{k}m}, \mathcal{R}_{time\_\overline{k}m}, \vec{\lambda}_{\overline{k}m})\},$$

   a random element at index $\hat{k} \in \{1, \ldots, m\}$ is extracted.
5. A new profile $\mathcal{P}_{output}$ is created:

$$\mathcal{P}_{output} = \mathcal{P}_{input} \cup \{(id_{\overline{k}}, a_{\overline{k}\hat{k}}, r_{\overline{k}\hat{k}}, \mathcal{R}_{time\_\overline{k}\hat{k}}, \vec{\lambda}_{\overline{k}\hat{k}})\} \setminus \{(id_{\overline{k}}, a_{\overline{k}}, r_{\overline{k}}, \mathcal{R}_{time\_\overline{k}}, \vec{\lambda}_{\overline{k}})\}.$$

6. Then $\mathcal{W}(\mathcal{P}_{output})$ is evaluated and $\mathcal{S} = \mathcal{S} \cup \{(\mathcal{P}_{output}, \mathcal{W}(\mathcal{P}_{output}))\}$ updated.
7. Then, $\mathcal{S}$ is sorted increasing by the values of Time-Workload Coefficient $\mathcal{W}$ of every Recommendation Profile.
8. Finally, the procedure is repeated using as input the profile with the lower Time-Workload coefficient between $\mathcal{P}_{input}$ and $\mathcal{P}_{output}$.
9. The procedure stops after n iterations in which $\mathcal{W}(\mathcal{P}_{input}) \leq \mathcal{W}(\mathcal{P}_{output})$.[3]

[3] In our implementation, n is set equal to 200.

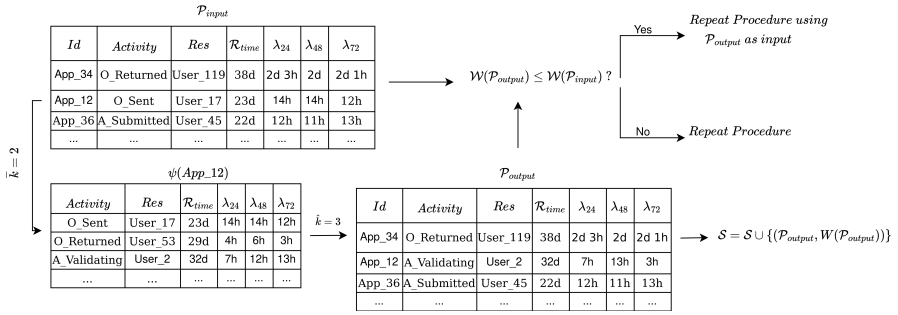


Fig. 3: Generation of Recommendations Profiles. The input Recommendation Profile $\mathcal{P}_{input}$ is in the top-left corner. The element with id = App_34 is randomly picked, and the associated output of ψ is represented below. From it, a random element is drawn, and a new Recommendation Profile $\mathcal{P}_{output}$ is generated, replacing the value with id = App_34 with them. Finally, the S is updated, and the procedure is iterated.

An example of the procedure is depicted in Figure 3. The profile $\mathcal{P}_{input}$ is initially taken as input, then, the 2^{nd} element $(App\_12, O\_Sent, User\_17, 23d, 14h, 14h, 12h)$ is randomly drawn and for the corresponding trace identifier App_12 the Experience Based Oracle $\psi(App\_34)$ is evaluated. Thereafter, the third element of the output of $\psi(App\_34)$, i.e. $(A\_Validating, User\_2, 32d, 7h, 12h, 13h)$, is randomly picked to generate the profile $\mathcal{P}_{output}$ replacing the values of the previous one. Finally, $\mathcal{P}_{output}$ is added to the sequence S, and the profile with the lower value of the Time-Workload Coefficient is used as input for another iteration of the same process.

4.3 Recommendation of Activity-Resource Pairs

Often, due to the possible organizational constraints, it is desirable not directly to pick the best profile but to return the top-k profiles from the sequence $\mathcal{S}$. Therefore, for every active trace in the set $\mathcal{I}$, the system proposes k alternatives for each trace identifier id_i in the set $\mathcal{I}$. We build a recommendation set $S_{id_i} = \{(a_{id_i\_1}, r_{id_i\_1}), \ldots, (a_{id_i\_k}, r_{id_i\_k})\}$, with k different recommendation pairs. The initial profile $\mathcal{P}_0 \in \mathcal{S}$ initiates the first pair in the recommendation sequence. Specifically, the elements of the first pair of every S_{id_i}, i.e. $(a_{id_i\_1}, r_{id_i\_1})$, are the activity a_i and the resource r_i from the corresponding element in $\mathcal{P}_0$, i.e. $(id_i, a_i, r_i, \mathcal{R}_{time_i}, \vec{\lambda}_i) \in \mathcal{P}_0$.

Then, for every profile $\mathcal{P}_j \in \mathcal{S}$, if the pair (a_i, r_i) in the element with identifier id_i, i.e. $(id_i, a_i, r_i, \mathcal{R}_{time\_i}, \vec{\lambda}_i) \in \mathcal{P}_j$ is not in S_{id_i}, it is added to the profile, if not, the j + 1-th element is checked, till the set of S_{id_i} has k elements. This procedure enables the recommendation of the best k pairs of activity and resource belonging to the Recommendation Profiles with the best Time-Workload coefficient.

The procedure starts by ranking traces in descending order of remaining time. Subsequently, it involves the utilization of functions λ and ψ to generate an initial recommendation profile $\mathcal{P}_{input}$. Subsequently, additional profiles are generated through an algorithm aimed at consistently improving profiles and minimizing the Time-Workload

Coefficient $\mathcal{W}$. Ultimately, for a given trace identifier, k distinct activity-resource pairs are allocated from the profile with the lowest $\mathcal{W}$.

5 Evaluation

We evaluate the efficacy of our approach in achieving a balanced resource allocation while improving process performance and compare it to a state-of-the-art benchmark system. In the benchmark system recommendations do not consider the resource experience, i.e., recommendations are solely based on assigning the most suitable activity to the most proficient resource without balancing resource utilization.

To evaluate the recommendations provided, we compare execution times obtained via simulations of the process and those times measured in the reality. Specifically, we leverage a Business Process Simulation (BPS) approach [8] to simulate process instances in which our recommendations are implemented and compare those with continuations of the traces in reality, i.e., from the event log.

5.1 Processes and Datasets

The approach has been evaluated with two different processes for which we could obtain suitable event logs with rich information on the resource perspective: *Bank Account Closure (BAC)*, a process of an Italian bank institution that deals with the closures of bank accounts; *BPIC17AO*, the subprocess for the application-relevant (A) and offer-relevant activities (O) in the 2017 BPI Challenge event data, a log of a loan application process from a Dutch financial institution.

Adams et al. [2] found a concept-drift in the BPIC17AO event log: an increase in the workload of resources at week 22 led to a decrease in the service times at week 28 [2]. Since, our approach assumes a process log without concept drift we partitioned the BPIC17AO event log into two sublogs: one containing traces before week 22, namely BPIC17AO-before, and the other containing those after the 28^{th} week, namely BPIC17AO-after. This allows us to experiment across three distinct datasets.

5.2 Experimental Setup

The approach has been implemented in Python using Catboost [25] for training the oracle and task duration functions.[4] The traces were encoded using the same procedure described in [23], with the time units required by the specified resource as the dependent variable, thereby formulating an equivalent regression problem. In line with common supervised learning practice, we divided the event log $\mathcal{L}$ extracting a training and a test event log, $\mathcal{L}^{comp}$ and $\mathcal{L}^{run}$, respectively. To extract the training log we compute the earliest time t_{split} such that 70% 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 test log $\mathcal{L}^{run}$ are truncated to a set $\mathcal{L}^{trunc}$, namely the set of prefixes, that is obtained from $\mathcal{L}^{run}$ by removing every event

[4] `https://github.com/Pado123/rbranch`

with a timestamp larger than t_{split}: $\mathcal{L}^{trunc}$ only contains the events occurred before time t_{split}. This procedure tries to mimic the reality at time t_{split} and was used in [23].

The Task Duration Function, and consequently the whole system, was trained on $\mathcal{L}^{comp}$, employing cross-validation for the tuning of the hyper-parameters.

We use $\mathcal{L}^{trunc}$ for generating the set of recommendations to be evaluated. Without access to the process for an A/B test, and since $\mathcal{L}^{run}$ does not consider the recommendation, we leverage a BPS model emulate the effect of recommendations to some of the system behavior that needs to be considered for a fair and accurate evaluation.

We implemented a simulator[5] based on a Petri net model of the process and a set of simulation parameters computed through analysis on the original event log:

Control-flow perspective. The Petri net models were discovered using the Inductive Miner, and later manually altered to increase the fitness.

Resource perspective. We identified the pool of resources grouping them into roles, and then, we determined the activities associated with each roles, leveraging on the technique proposed by Burattin et al. in [6].

Working calendars. We assigned the roles working schedules by analyzing the daily hours and the weekly days in which each role performs some activity.

Inter-arrival time. We computed the inter-arrival times between process executions, i.e., the differences between the start timestamps of two subsequent traces. Then, comparing with various probability distributions, we find the most suitable distribution, retaining the one that minimizes the Earth Mover Distance metric [8].

Activity-duration distributions. For each process' activity and allowed resource, we used the event log to determine the best fitting distribution of the activity durations.

Branching probabilities. Employing the technique proposed by de Leoni et al. [20], we computed transition weights used for determining the likelihood of executing an activity at a process state.

By focusing on different process perspectives (resource, temporal and control flow), we aim to obtain accurate and sufficiently precise simulations for fair evaluation.

The simulator is initialized by setting the resource work queues at time t_{split} as observed for the process in the event log, to mimic the actual state at timestamp t_{split}. In particular, the simulation model was used for each trace $\sigma \in \mathcal{L}^{trunc}$ to evaluate the effect of the recommendation. The trace σ was replayed on the simulation model, so as to take the process state (namely the marking) to that after σ: for the reached state, the recommendation was made occur in the simulation model and hence replayed, and then the continuation till the final state was simulated for a statistically significant and sufficient number of times. As mentioned in Section 4, the potential activity recommendations were ranked, and the highest ranked recommendation allowed by the simulation model was made occur. In fact, the system might potentially - although seldom - recommend to perform an activity that does not make sense in reality (at least as far as the simulation model goes). Even if very rarely, it might be the case that no recommendation is allowed by the simulation model: in this case, no recommendation was followed by the process' running instance under analysis.

[5] `https://github.com/franvinci/RecsSysBPSEvaluator`

As mentioned above, we decide to maximize the fitness to minimize the chances for a recommendation not to be replayable by the model. In fact, the Petri-net models for the BPIC17AO-before and BPIC17AO-after process, which are identical (the concept drift refers to the resource and time perspectives), reach a score of 0.88 in fitness, where the BAC process Petri net model has even a fitness of 0.96. Of course, fitness and precision are oftentimes contrasting: indeed, precision was 0.61 and 0.84. This however poses very limited consequences on the simulation-model accuracy, because the infrequent branches, which cause a precision reduction, would also be associated with low probabilities of occurring, proportionally to the frequency.

Ultimately, we conducted simulations of process instances incorporating these recommendations, resulting in individual process traces for each prefix, obtaining a set of simulated traces $\mathcal{L}^{sim}$. Subsequently, we compared the performance metrics derived from these simulations with those obtained from real-world measurements.

5.3 Experimental Results

Given the set of test prefixes $\mathcal{L}^{trunc}$ at the split time t_{split}, we computed the time required to conclude each prefix $p \in \mathcal{L}^{trunc}$ using simulations. In fact, our recommendations can only influence events occurring after t_{split}. Hereafter, we define the trace obtained via simulation from the prefix $p \in \mathcal{L}^{trunc}$ as $\sigma_p^{sim} \in \mathcal{L}^{sim}$, while the trace observed in reality as $\sigma_p^{real} \in \mathcal{L}^{run}$. We computed the remaining time $\mathcal{R}_{time}(\sigma_p^{sim}, t_{split})$, as defined in Section 3, and we compare it with $\mathcal{R}_{time}(\sigma_p^{real}, t_{split})$ for each prefix $p \in \mathcal{L}^{trunc}$. Particularly, the comparison is based on the difference between the values observed in reality and those obtained via simulation for each prefix p, i.e. $\mathcal{R}_{time}(\sigma_p^{real}, t_{split}) - \mathcal{R}_{time}(\sigma_p^{sim}, t_{split})$. Therefore, positive values in these comparisons indicate the improvement due to adopting recommendations, in fact our goal is to minimize the remaining time. Finally, the evaluation is based on the global improvement computed as the sum of improvements for each test trace in terms of time units, i.e. $\sum_{p \in \mathcal{L}^{trunc}} \big(\mathcal{R}_{time}(\sigma_p^{real}, t_{split}) - \mathcal{R}_{time}(\sigma_p^{sim}, t_{split})\big)$.

We also assess the relative improvement wrt. the remaining time observed in the test log is computed as the ratio between the total improvement and the sum of the remaining times observed in reality.

To address the stochasticity inherent in simulations, we conducted simulations ten times for each of the three processes. Table 2 shows the final results. The reported values represent the average results obtained across each simulation, with their respective standard deviations in the parenthesis.

Table 2 also reports on the percentage of feasible recommendations within the simulation model, as discussed in Section 5.2. We compare the results obtained by our technique, labelled as *Workload Distribution*, with those obtained by a benchmark method, labelled as *Benchmark*, where recommendations are assigning the most suitable activity to the most proficient resource, without regarding for balanced resource utilization.

The results show that our approach leads to better performance, decreasing the remaining time to complete a trace with respect to what happened in the reality. We outperform the benchmark in 2 out of 3 cases and results show that always assigning tasks to the best performing resource can result in long queues and increased remaining times.

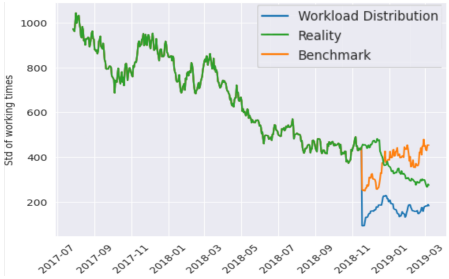
Dataset	Method	Tot. Impr.	Relative Tot. Impr.	Feasible Recs
BAC	Workload Dist.	5028.96 h (1150.18 h)	0.07 (0.02)	95.63%
	Benchmark	2156.77 h (2236.04 h)	0.03 (0.04)	96.03%
BPIC17AO-before	Workload Dist.	123050.14 h (5841.34 h)	0.36 (0.02)	95.69%
	Benchmark	111754.49 h (4117.23 h)	0.33 (0.01)	81.43%
BPIC17AO-after	Workload Dist.	20568.03 h (5058.91 h)	0.06 (0.01)	90.71%
	Benchmark	82575.09 h (6317.67 h)	0.24 (0.02)	72.55%

Table 2: Total improvement (in *hours*) and relative total improvement averaged over ten simulations comparing our approach (Workload Distribution) to the benchmark approach (Benchmark). The standard deviation is reported in parenthesis and the percentage of feasible recommendations for the simulation is shown in the last column.

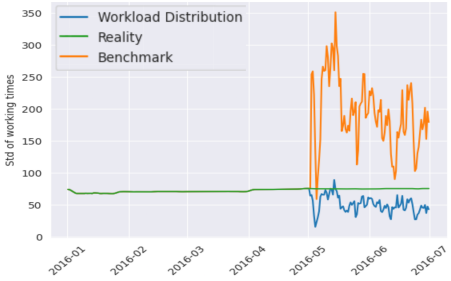
In the BPIC17AO-after dataset, our recommendation system shows less improvement compared to the benchmark. The variation in outcomes compared to BPIC17AO-before is probably due to the concept-drift, since a difference in the workload of resources has been observed [2]. However, our method provides more suitable recommendations, showing an higher percentage of feasible recommendations compared to those obtained from the benchmark one. In fact, recommending the most efficient activity performed by the most efficient resource can lead to recommendations that might be optimal, but not feasible in practice: e.g. a resource may not execute some task, or an activity may not be performed in a certain process state.

Figures 4 show the rolling average standard deviation of cumulative resources working times (in *hours*), i.e. the time spent performing tasks. Lower values indicate a better balancing in resource allocation. In fact, a low value means that the deviation from the mean for the resource working times are similar, and hence, the resources have similar working times. The results align with our goal to achieve a better balance in the resource utilization. Moreover, while our approach shows a relatively lower improvement for the BPIC17AO-after dataset compared to the benchmark, Figure 4c reveals that using our recommendations leads to enhanced balance and stable resource utilization. BPIC17AO datasets show a constant standard deviation of cumulative working hours across resources per timestamp, indicating no alterations in resource utilization during that time frame. However, we can notice higher values for the BPIC17AO-after dataset attributed to the workload of resources concept drift [2]. Moreover, shifts observed at the time split using recommendations are due to the sudden changes in them.

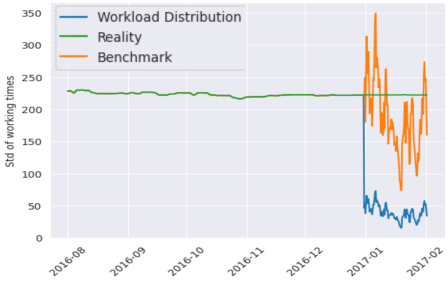
Table 3 presents the average coefficient of variation for each activity based on how many times each resource executed it. The coefficient of variation was selected to ensure comparability across different logs, as using the standard deviation would be influenced by the differing number of traces. The findings clearly indicate that the benchmark approach improves workload distribution by balancing the frequency with which a resource executes specific tasks.



(a) BAC dataset.



(b) BPIC17AO-before dataset.



(c) BPIC17AO-after dataset.

Fig. 4: Rolling average standard deviation of cumulative resource working times (*hours*) over ten simulations. Lower values indicate a better balancing in the resource utilization.

Case Study	Reality	Benchmark	Workload Distribution
BAC	1.31	0.50	**0.43**
BPIC17AO-before	1.72	0.57	**0.45**
BPIC17AO-after	1.72	0.51	**0.45**

Table 3: Average coefficient of variation computed for each activity based on how many times the different resources execute that activity. Lower values indicate a more balanced workload distribution among resources.

6 Conclusion

Recommending the next activity and the best-fitting resource for a running process instance to reduce the remaining time until completion is a core task in prescriptive process analytics. The approach proposed in this paper focuses on improving the remaining time and takes into account resources' experience and workload when recommending a suitable resource allocation for cases in which an intervention is required. Compared to a benchmark approach, which only allocates resource based on predicted performance, our experiments on event data from two real-life processes on simulation models indicate that the proposed approach leads to a more balanced workload that typically translate to a lower overall remaining time. Beyond an overall improvement of the performance indicator, our work can also be seen through the lens of fairness towards individual resources. Resources should be neither overloaded nor never chosen. Tasks

are allocated tasks based on the idea to reduce the standard deviation of time it takes any resource to perform and activity and distributing workload. Future work could explore applying the proposed approach using other performance indicators, such as total cost.

As most work on prescriptive process analysis, we could not use an A/B test for our evaluation. This limits the validity of our results based on the accuracy of the simulation model. In the future, we envision to use the top-k recommendations as a catalogue of tasks to choose as next for the resources leaving them the autonomy to select tasks as in [22] while being still optimized with respect to resource experience and workload distribution, accommodating organizational flexibility and efficiency in task assignment.

Acknowledgements. A large share of this work was conducted during a visit of Mr. Padella at TU/e, which was partly supported by EU through the Erasmus-Mundus BDMA program. The work of Dr. Mannhardt was partially supported by Smart Journey Mining, a project funded by the Research Council of Norway (grant no. 312198).

References

1. van der Aalst, W.M.P.: Responsible data science: Using event data in a "people friendly" manner. In: 18th International Conference on Enterprise Information Systems, ICEIS 2016, Proceedings (2016)
2. Adams, J.N., van Zelst, S.J., Quack, L., Hausmann, K., van der Aalst, W.M.P., Rose, T.: A framework for explainable concept drift detection in process mining. In: 19th International Conference on Business Process Management, BPM 2021, Proceedings (2021)
3. Arias, M., Munoz-Gama, J., Sepúlveda, M.: Towards a taxonomy of human resource allocation criteria. In: 16th International Conference on Business Process Management, BPM 2018, Proceedings (2018)
4. Arias, M., Rojas, E., Munoz-Gama, J., Sepúlveda, M.: A framework for recommending resource allocation based on process mining. In: 13th International Conference on Business Process Management, BPM 2015, Proceedings (2015)
5. Bidar, R., ter Hofstede, A., Sindhgatta, R., Ouyang, C.: Preference-based resource and task allocation in business process automation. In: On the Move to Meaningful Internet Systems: OTM 2019 Conferences, Proceedings (2019)
6. Burattin, A., Sperduti, A., Veluscek, M.: Business models enhancement through discovery of roles. In: 2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), Proceedings (2013)
7. Cabanillas, C., García, J., Resinas, M., Ruiz, D., Mendling, J., Ruiz-Cortés, A.: Priority-based human resource allocation in business processes. Service-Oriented Computing (2013)
8. Chapela-Campa, D., Benchekroun, I., Baron, O., Dumas, M., Krass, D., Senderovich, A.: Can i trust my simulation model? measuring the quality of business process simulation models. In: 21th International Conference on Business Process Management, BPM 2023, Proceedings (2023)
9. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.: Fundamentals of business process management. Springer (2018)
10. Erasmus, J., Vanderfeesten, I., Traganos, K., Jie-A-Looi, X., Kleingeld, A., Grefen, P.: A method to enable ability-based human resource allocation in business process management systems. In: The Practice of Enterprise Modeling (POEM), Proceedings (2018)
11. Goel, K., Fehrer, T., Röglinger, M., Wynn, M.T.: Not here, but there: Human resource allocation patterns. In: 21th Business Process Management Conference, BPM 2023, Proceedings (2023)

12. Huang, Z., Lu, X., Duan, H.: Mining association rules to support resource allocation in business process management. Expert Systems with Applications **38** (2011)
13. Huang, Z., Lu, X., Duan, H.: Resource behavior measure and application in business process management. Expert Systems with Applications **39** (2012)
14. Ihde, S., Pufahl, L., Völker, M., Goel, A., Weske, M.: A framework for modeling and executing task-specific resource allocations in business processes. Computing **104** (2022)
15. Kabicher-Fuchs, S., Mangler, J., Rinderle-Ma, S.: Experience breeding in process-aware information systems. In: 25th International Conference in Advanced Information Systems Engineering, CAiSE 2013 (2013)
16. Kabicher-Fuchs, S., Rinderle-Ma, S.: Work experience in pais – concepts, measurements and potentials. In: 24th International Conference in Advanced Information Systems Engineering, CAiSE 2012, Proceedings (2012)
17. Kim, J., Comuzzi, M., Dumas, M., Maggi, F.M., Teinemaa, I.: Encoding resource experience for predictive process monitoring. Decision Support Systems **153** (2022)
18. Kubrak, K., Milani, F., Nolte, A., Dumas, M.: Prescriptive process monitoring: Quo vadis? PeerJ Computer Science (2022)
19. Kumar, A., Dijkman, R., Song, M.: Optimal resource assignment in workflows for maximizing cooperation. In: 11th International Conference on Business Process Management, BPM 2013, Proceedings (2013)
20. de Leoni, M., Vinci, F., Leemans, S.J.J., Mannhardt, F.: Investigating the influence of data-aware process states on activity probabilities in simulation models: Does accuracy improve? In: 21th International Conference on Business Process Management, BPM 2023, Proceedings (2023)
21. Ly, L.T., Rinderle, S., Dadam, P., Reichert, M.: Mining staff assignment rules from event-based data. In: 3rd International Conference on Business Process Management, BPM 2005 Workshops, Proceedings (2005)
22. Padella, A., de Leoni, M.: Resource allocation in recommender systems for global kpi improvement. In: 21th International Conference on Business Process Management, BPM 2023, Proceedings (2023)
23. Padella, A., de Leoni, M., Dogan, O., Galanti, R.: Explainable process prescriptive analytics. In: 4th International Conference on Process Mining, ICPM 2022 (2022)
24. Pika, A., Leyer, M., Wynn, M.T., Fidge, C.J., ter Hofstede, A.H.M., van der Aalst, W.M.P.: Mining resource profiles from event logs. ACM Transactions on Management Information Systems (2017)
25. Prokhorenkova, L.O., Gusev, G., Vorobev, A., Dorogush, A.V., Gulin, A.: Catboost: unbiased boosting with categorical features. In: NeurIPS (2018)
26. Pufahl, L., Ihde, S., Stiehle, F., Weske, M., Weber, I.: Automatic resource allocation in business processes: A systematic literature survey. ArXiv (2021)
27. Reijers, H.A., Mansar, S.L.: Best practices in business process redesign: an overview and qualitative evaluation of successful redesign heuristics. Omega **33**(4) (2005)
28. Shoush, M., Dumas, M.: When to intervene? prescriptive process monitoring under uncertainty and resource constraints. In: 20th International Conference on Business Process Management, BPM 2022, Proceedings (2022)
29. Vanderfeesten, I.T.P., Grefen, P.: Advanced dynamic role resolution in business processes. In: 27th Conference on Advanced Information Systems Engineering (CAiSE) Workshops, Proceedings (2015)
30. Zhao, W., Pu, S., Jiang, D.: A human resource allocation method for business processes using team faultlines. Applied Intelligence (2020)
31. Zhao, W., Yang, L., Liu, H., Wu, R.: The optimization of resource allocation based on process mining. In: Advanced Intelligent Computing Theories and Applications (2015)