Event Data-Driven Feasibility Checking of Process Schedules

Hannes Häfke[1] and Sebastiaan J. van Zelst[1,2]

[1] Fraunhofer Institute for Applied Information Technology, Sankt Augustin, Germany
{hannes.haefke, sebastiaan.van.zelst}@fit.fraunhofer.de
[2] RWTH Aachen University, Aachen, Germany

Abstract. Numerous processes require dedicated scheduling of their to-be-executed activities. Various algorithms have been developed to computationally solve many different scheduling problems, allocating the available resources to predefined time slots of activity execution to (theoretically) maximize resource utilization efficiency. Yet, in industry, creating schedules for future process executions often remains a (primarily) manual, spreadsheet-based endeavor. Typically, manually created schedules are sub-optimal and potentially infeasible. At the same time, the event data stored in the information systems supporting the process can act as valuable input to further improve the general alignment of the schedule to the actual process execution. Therefore, in this paper, we propose a novel method that enables schedule feasibility checking based on historically recorded event data corresponding to the actual execution of the scheduled process. Our method serves as an input to detect significant issues in the project scheduling problems, which can be used to further improve the overall quality of the schedules computed. Our initial results confirm the general applicability of the proposed framework.

Keywords: Business Process Management · Process Mining · Data-driven Scheduling · Business Process Organization

1 Introduction

The efficient execution of business processes includes decision-making from various perspectives to achieve the desired business outcomes. One such perspective is *scheduling* [20], i.e., a sub-field of the broader field of Operations Research (OR) [24,23], concerned with the assignment of to-be-executed activities to available resources at a predefined designated time-slot. It is generally recognized that accurate scheduling is vital in many areas, e.g., in production and service industries, to ensure a competitive advantage. Hence, several scheduling algorithms exist that allow for computationally solving many different scheduling problems, i.e., either yielding an optimal solution (defined in terms of some desirable outcome of the schedule, e.g., overall lead time) or an approximate solution.

Albeit numerous classes of scheduling problems and their associated solution approaches have been investigated in tremendous depth, creating schedules

in the industry remains a largely manual, spreadsheet-based endeavor. In [6], two main causes are presented for the lack of adoption: (i) Advanced Planning Systems (APS) are considered a black box, and (ii) financial gains of APS adoption are unclear. However, the advantages of APS adoption are numerous, e.g., reduced manual labor in planning, varying optimization criteria, reduced errors, etc. At the same time, the information systems supporting scheduled processes, e.g., Enterprise Resource Planning (ERP) and Manufacturing Execution Systems (MES), track the historical process execution in great detail. These recorded data, i.e., referred to as *event data*, provide a valuable source of, arguably, objective evidence of the actual execution performance of the company. Said event data can, in turn, be used to detect problems in a schedule (created either manually or by an APS) and recommend alternative resource allocations.

Scheduling and the structured (semi-automated) analysis of event data (i.e., *process mining* [1]) have both been intensively studied. Yet, work focusing on their intersection remains scarce. Some work focuses on resource allocation [2,4] or combining event data with queueing theory [22,5]. However, none of these techniques allow for improving operational schedules based on historical behavior. Yet, at the same time, it is clear that historical event data is a precious resource in increasing the overall quality of the application of scheduling in practice. Therefore, this paper presents a novel approach for assessing scheduling feasibility based on event data. Our approach learns a statistical characterization of the past performance of resources, which it subsequently uses to detect potential infeasibilities in the given schedule. To the best of our knowledge, our work is the first to seek to connect the operational history of a process with an organization's scheduling function.

We present an initial evaluation of our approach using real event data combined with artificially generated corresponding schedules. We assess the parameter sensitivity of two instantiations of our framework and show that our approach is computationally feasible. Furthermore, we investigate typical distributions that tend to fit concerning the training event data.

The remainder of this paper is structured as follows. In Section 2, we present related work. In Section 3, we present key background concepts. Section 4 presents our newly developed framework. In Section 5, we evaluate our approach. Section 6 concludes this work.

2 Related Work

Ample work exists on scheduling, i.e., covering scheduling theory, algorithms, models, systems, etc. [20,13]. Most work in scheduling research is algorithm oriented and focuses on static scheduling problems. A smaller subset focuses on the notion of *dynamic scheduling* [18], i.e., an ongoing reactive scheduling process in which real-time events and information forces schedule reconsideration.

A limited amount of work has considered the intersection of process-generated event data and scheduling. In [16], a revised WfMS implementation is presented that supports both *flow* and *schedule tasks*, which allows integration of em-

ployee calendars within the task-scheduling. In [17, Section 11.3], process mining algorithms are used to identify future task scheduling, based on event logs containing (historical) appointment information. In [15], the authors propose event-log-based visualization techniques to visualize changes in the process execution, primarily focusing on the resource and time dimension. Whereas the framework is generic, the proposed instantiation heavily focuses on the resource and temporal dimension, allowing the analysis of the impact of alternative work schedules. In [8], a scheduling approach is proposed for workforce scheduling. The authors do not exploit recorded event data, yet, illustrate how the inputs of the proposed scheduling problem relate to the BPM discipline,, e.g., covering the *organizational perspective*, the *control-flow perspective*, etc. In [21], the authors propose to learn static scheduling models from event data. The approach learns a timed Petri net from the event data which is converted into a constraint programming problem. In [12], the authors propose to embed process mining techniques in the context of scheduling. However, the integration of process mining is on the output side, i.e., the solutions of an optimization problem are converted into synthetic event data which are used for the analysis of the proposed solution. Another line of work aims to transform event data into Gantt charts (rather than process models) for further analysis [3].

Some authors have considered using queueing models to explicitly model timing-related aspects of business processes. In [22], Senderovich et al. propose to convert an event log and a schedule of a process into a *Fork/Join Queuing Network*, which are subsequently compared to assess conformance and performance aspects of the schedule. In [5], the authors propose to estimate missing lifecycle data for the purpose of performance analysis based on queueing models.

Several authors have focused on exploiting event data in the context of (human) resource allocation, i.e., in the context of business process executions. For example, in [2], Arias et al. propose a framework that exploits contextual information together with event data to improve human resource allocation. In [4], the authors propose to learn, i.e., based on event data, which resource allocations may lead to execution problems and propose a resource selection mechanism that minimizes the risk of execution problems when executing future tasks. In [11], Ihde et al. focus on a software design for a resource-aware task allocation service that can replace existing task schedulers in BPM systems. Havur et al. [9,10] propose a resource allocation mechanism based on answer set programming that is able to handle resource dependencies and conflicts.

Whereas the works mentioned consider event data and schedules/resource allocation, to the best of our knowledge, our work is the first to seek to connect the operational history of a process with an organization's scheduling function.

3 Preliminaries

In this section, we present the basic background concepts used in this paper. In Section 3.1, we briefly present the notation used. In Section 3.2, we formally define the notion of schedules. Finally, in Section 3.3, we present event data.

3.1 Notation

$\mathbb{Z}$ denotes the set of integers and $\mathbb{N}$ denotes the set of natural numbers including 0. We let $\mathbb{R}$ denote the set of real-valued numbers and we let $\mathbb{R}^+=\{x\in\mathbb{R}|x\geq 0\}$ denote the non-negative real-valued numbers. A *timestamp* $t\in\mathbb{R}^+$ is a point in time.[1] An *interval* is a set of real-valued numbers containing all numbers that lie between its two boundaries. Let $a, b\in\mathbb{R}$, the *closed interval* between a and b is defined as $[a, b]=\{x\in\mathbb{R}|a\leq x\leq b\}$. Observe that a closed interval can be empty (i.e., if $b>a$) and it can be a singleton set (i.e., if $a=b$), referred to as a *degenerate interval*. Any interval that is non-empty and not degenerate is a *proper interval*.

We assume that the reader is reasonably familiar with *probability theory* [14]. We particularly focus on *continuous probability functions*, with an associated *probability density function* f, e.g., the *normal distribution* [19] has density function $f(x)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$ for average value μ and standard deviation σ. Various methods exist to *fit* a probability distribution. Generally, given some $X\subset\mathbb{R}$, we let $\hat{f}=\theta(X)$ denote the result of applying a probability distribution fitting algorithm θ applied on X, where $\hat{f}$ represents the fitted distribution. Various different methods exist to assess the *goodness of fit* of $\hat{f}$ w.r.t. X. We generally let ϵ denote the error of $\hat{f}$ with respect to X.

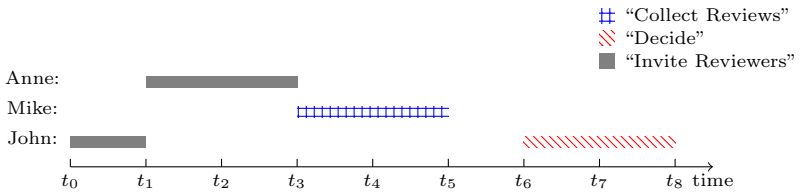
3.2 Schedules

In the context of this paper, we formalize the notion of a schedule. Conceptually, a schedule describes a collection of *tasks* that describe that some *activity* is intended to be executed by a *resource* within a predefined *time-frame*. For example, consider Fig. 1, in which we present visual examples of two schedules of an academic reviewing process. In the schedule in Fig. 1a, *John* is scheduled to "Collect Reviews" in time window $[t_0, t_1]$, and activity "Decide" in time window $[t_6, t_8]$. *Mike* is scheduled to "Invite Reviewers" in time window $[t_3, t_5]$. *Anne* is scheduled to "Collect Reviews" in time window $[t_1, t_3]$. In the example schedule in Fig. 1b, the same activities are scheduled to the same set of resources. However, in the schedule, there is no obsolete idle time in between any tasks, and, the tasks are scheduled as early as possible (e.g., Anne starts inviting reviewers directly at t_0, reviews are directly collected after reviewer invitation is completed, etc.). As such, schedule Fig. 1b is expected to finish earlier. In the context of this paper, we define the notion of a *task* and corresponding as follows.

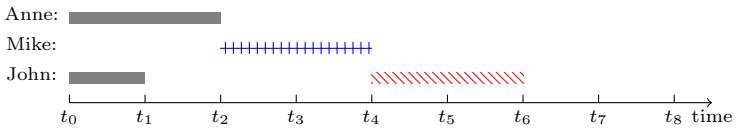
Definition 1 (Task; Schedule). *Let Σ denote the universe of business activities and let $\mathcal{R}$ denote the universe of resources. A task τ is a tuple $\tau=(a, r, t_1, t_2)\in\Sigma\times\mathcal{R}\times\mathbb{R}^+\times\mathbb{R}^+$, s.t. $t_2>t_1$. We let $\mathcal{T}$ denote the universe of tasks. A schedule is a set of tasks, i.e., $S\subseteq\mathcal{T}$. We let $\mathcal{S}$ denote the universe of schedules.*

Observe that, for a given task $\tau=(a, r, t_1, t_2)$, the intended timeframe of execution is the interval $[t_1, t_2]$. Since $t_2>t_1$, any task is assumed to be scheduled in a *proper interval*. We let $\Delta(\tau)=[t_1, t_2]$ denote said interval. Reconsider

[1] We assume the existence of some minimal timestamp $t_0\in\mathbb{R}^+$ and $t'\in\mathbb{R}^+$ s.t. any timestamp t can be expressed as $t=t_0+t'$.



(a) Example of a *non-optimal schedule* of an academic reviewing process. *John* is scheduled to "Invite Reviewers" in time window $[t_0, t_1]$, and activity "Decide" in time window $[t_6, t_8]$. *Mike* is scheduled to "Collect Reviews" in time window $[t_3, t_5]$. *Anne* is scheduled to "Invite Reviewers" in time window $[t_1, t_3]$.



(b) Example of an *optimal schedule* of an academic reviewing process. *John* is scheduled to "Invite Reviewers" in time window $[t_0, t_1]$, and activity "Decide" in time window $[t_4, t_6]$. *Mike* is scheduled to "Collect Reviews" in time window $[t_2, t_4]$. *Anne* is scheduled to "Invite Reviewers" in time window $[t_0, t_2]$.

Fig. 1: Two example schedules. The schedule in Fig. 1a is non-optimal, the schedule in Fig. 1b is optimal and is expected to finish earlier.

Fig. 1a, which can be written as $\{(\text{"Invite Reviewers"}, \text{John}, t_0, t_1), (\text{"Decide"}, \text{John}, t_6, t_8), (\text{"Collect Reviews"}, \text{Mike}, t_3, t_5), (\text{"Invite Reviewers"}, \text{Anne}, t_2, t_6)\}$.

Various *constraints* can render a schedule (in)feasible. For example, if we only require a resource to work on one task at a time, both schedules in Fig. 1 are feasible. However, suppose we require a minimal setup time of four time-units (idle time in-between two consecutive activities executed by a resource). In that case, the example schedule in Fig. 1b is not feasible (John only has three time-units of setup time between his two activities). Finally, a strict subset of all feasible schedules, i.e., given a set of constraints, is *optimal*. Optimality of a schedule is determined by an *objective function*, i.e., a function that needs to be minimized or maximized. For example, assuming that no setup times are required, if we aim to optimize the *makespan* (minimizing the end time of any task), the schedule in Fig. 1a is not optimal whereas Fig. 1b is (assuming that the schedule needs to adhere to the general control-flow of academic reviewing). A schedule may be optimal under a particular set of constraints yet infeasible for a slightly different set of constraints. In the context of this paper, we primarily assume that a given schedule is *feasible*, yet, it may be sub-optimal.

3.3 Event Data

The information systems supporting business processes, e.g., Enterprise Resource Planning (ERP) systems, track the execution of the different activities executed,

Table 1: Event data required for scheduling

Event ID	*Case ID*	*Task*	*Start Time*	*End Time*	*Resource*
⋮	⋮	⋮	⋮	⋮	⋮
1337	331	Invite reviewers	01/15/2009 15:40	01/15/2009 15:59	Mike
1338	331	Get review	02/19/2009 07:30	02/19/2009 15:22	Carol
1339	332	Invite reviewers	05/15/2009 12:10	05/15/2009 13:01	Anne
⋮	⋮	⋮	⋮	⋮	⋮

i.e., referred to as an *event log*. Consider Table 1, in which we present a simplified example of an event log. Each row in the table describes an *activity instance*, i.e., a historical recording of the execution of an activity. The first row represents a recording of the "Invite reviewers" activity. The activity instance has a *unique identifier*, i.e., 1337. Similarly, the activity instance has a unique *case identifier*, i.e., 331, representing the process instance for which the activity was executed. Two *timestamps* are recorded for the activity instance, i.e., a *start* and *end timestamp*. Finally, the activity instance records which *resource* executed the activity. Generally, additional data attributes may be available for activity instances, e.g., a customer ID, product ID, or associated costs. However, for simplicity, we only focus on the data attributes strictly required for our approach.[2] We formalize the notion of event data as follows.

Definition 2 (Activity Instance, Event Log). *Let Σ denote the universe of activity labels and let $\mathcal{R}$ denote the universe of resources. An* activity instance*, i.e., a tuple $\upsilon{=}(i, c, a, r, t_1, t_2){\in}\mathbb{N}{\times}\mathbb{N}{\times}\Sigma{\times}\mathcal{R}{\times}\mathbb{R}^+{\times}\mathbb{R}^+$, describes the historical recording of an activity a, executed by resource r during time-frame $[t_1, t_2]$ ($t_1{<}t_2$), in the context of case c. Attribute i represents the activity instance's* unique identifier*. We let $\mathcal{I}$ denote the universe of activity instances, i.e., for any $\upsilon, \upsilon'{\in}\mathcal{I}$ with $\upsilon{=}(i, c, a, r, t_1, t_2)$, $\upsilon'{=}(i', c', a', r', t'_1, t'_2)$, if $i{=}i'$ then $\upsilon{=}\upsilon'$.*

An event log *L is a collection of activity instances, i.e., $L{\subseteq}\mathcal{I}$.*

Let $\upsilon{=}(i, c, a, r, t_1, t_2)$ be an activity instance. Similarly to tasks, we let $\Delta(\upsilon){=}[t_1, t_2]$, yet, $\Delta(\upsilon)$ describes the *actual time interval* in which the activity instance was observed (opposed to the scheduled/expected time).

4 Event-Data-Driven Feasibility Checking

In this section, we present our main contribution, i.e., *event-data-driven feasibility checking of schedules*. We present a general overview of our approach in Section 4.1. In Section 4.2, we present a generic definition and two instantiations of *duration estimators*, i.e., data-driven statistics to estimate the duration of a task executed by a resource. Finally, in Section 4.3, we describe the general mechanism to revise a given schedule based on the learned estimators to be subsequently used for feasibility checking.

[2] Unlike most process mining works, we do not explicitly require the presence of a case identifier. However, Definition 2 follows the conventional definition.

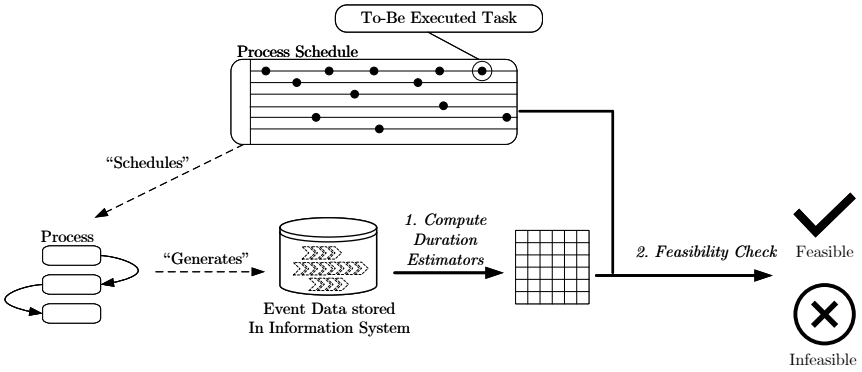


Fig. 2: Schematic overview of the proposed approach. The proposed approach consists of two steps, i.e., *1. learning an activity-resource statistics matrix* and *2. Checking schedule feasibility*

4.1 Overview

In this section, we briefly present an overview of our proposed framework. Consider Fig. 2, in which we present a schematic overview. Our framework consists of two main steps. In the first step, from the input event log, we derive a *duration estimator*. We use the event data to record different statistics (e.g., averages, distributions, etc.) of the historical execution performance of an activity-resource combination. In the second step, we perform a feasibility check of a given theoretically feasible schedule. The core idea of the feasibility check is to find scheduled tasks that, according to the data-based duration estimator, are either *overly optimistic or overly pessimistic*, i.e., in terms of their expected duration. Suppose we detect such a task in the schedule. In that case, we use the activity-resource statistic to compute a more reasonable expected duration, i.e., we use a statistic based on historical $\Delta(\upsilon)$ values to revise $\Delta(\tau)$ values (for $\upsilon \in \mathcal{I}$ and $\tau \in \mathcal{T}$). We subsequently replace the $\Delta(\tau)$ values in the given schedule with a more realistic data-driven timeframe $[t'_1, t'_2]$ and assess if the schedule remains feasible. We generally assume that the underlying scheduling problem is known, i.e., used to generate the schedule, or, that domain knowledge regarding general schedule feasibility is known. For example, there may be working hours in which tasks must be scheduled, mandatory setup times between two tasks, breaks, etc. We additionally assume that checking the feasibility of a given schedule w.r.t. a set of constraints is (programmatically) handled outside of our approach.

4.2 Deriving Activity-Resource Duration Estimators

In this section, we define the notion of duration estimators. We aim to compute an estimator for the expected duration of an activity $a \in \Sigma$ executed by a resource

$r \in \mathcal{R}$. For example, if we assume that event 1337 in Table 1 is representative for the performance of *Mike* for the activity "Invite reviewers", we assign an estimated value of 19 minutes. We first define the general notion of duration estimator, after which we present two corresponding instantiations.

Definition 3 (Duration Estimator). *Let Σ denote the universe of activities and let $\mathcal{R}$ denote the universe of resources. A duration estimator is a function $\hat{\Delta}\colon \Sigma \times \mathcal{R} \to \mathbb{R}^+$ estimating the expected duration of task execution.*

Given a task $\tau=(a, r, t_1, t_2)$ in some schedule S, for simplicity, we let $\hat{\Delta}(\tau)=\hat{\Delta}(a, r)$. Observe that $\tau'=(a, r, t_1, t_1+\hat{\Delta}(\tau))$ is a derived task, i.e., based on τ, starting at the same timestamp as τ, with the estimated duration for the activity and resource described by τ.

In the remainder, we propose two concrete instantiations for the duration estimator, i.e., an estimator based on the *empirical average* and a *distribution*-based duration estimator.

The empirical-average-based estimator uses the measured average duration of a task executed by a resource. Additionally, the standard deviation is added a number of $k \in \mathbb{N}$ times to the measured average. We formalize the empirical-average-based estimator as follows.

Definition 4 (Empirical Average Estimator). *Let $L \subseteq \mathcal{I}$ be an event log and let $k \in \mathbb{Z}$. The* empirical average duration estimator $\hat{\Delta}_{avg,k}\colon \Sigma \times \mathcal{R} \to \mathbb{R}^+$ *is a duration estimation function with:*

$$\hat{\Delta}_{avg,k}(a, r) = \overline{x} + k \cdot \sigma_{\overline{x}} \tag{1}$$

where $X=\{\Delta(v) | v \in L \wedge v=(i, c, a, r, t_1, t_2)\}$, $\overline{x}=\frac{\sum_{x \in X} x}{|X|}$, *and* $\sigma_{\overline{x}}=\sqrt{\frac{\sum_{x \in X}(x-\overline{x})}{|X|-1}}$

The empirical-average-based estimator ignores the underlying distribution of the duration of the activity executed by the resource. Furthermore, parameter k allows us to tune the degree of under or overestimation of the estimator w.r.t. the data, i.e., a negative value for k yields a low estimated value, a positive value yields a high value.

Additionally, we propose a distribution-based estimator. In the estimator, we fit a number of different probability distributions and select the best fitting value. Subsequently, we use the k-th percentile of the fitted probability distribution as an estimator.

Definition 5 (Distribution Estimator). *Let $L \subseteq \mathcal{I}$ be an event log and let $k \in [0, 1]$. The* distribution-based duration estimator $\hat{\Delta}_{dist,\hat{f},k}\colon \Sigma \times \mathcal{R} \to \mathbb{R}^+$ *is a duration estimation function with:*

$$\hat{\Delta}_{dist,\hat{f},k}(a, r) = b, \; s.t. \int_{-\infty}^{b} \hat{f}(x)dx = k \tag{2}$$

where $X=\{\Delta(v) | v \in L \wedge v=(i, c, a, r, t_1, t_2)\}$ *and* $\hat{f}=\theta(X)$ *is some* fitted continuous probability function.

Observe that, for k=0.1, we obtain the 10-th percentile of the cumulative distribution of $\hat{f}$ as an estimator. In practice, when computing $\hat{f}=\theta(X)$, we fit a number of different distributions based on X, e.g., the log-normal distribution, the Gamma distribution, etc. We pick the fitted distribution that minimizes the corresponding error ϵ, e.g., by using the *residual sum of squares*.

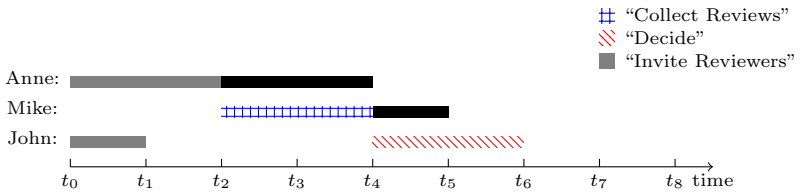
Observe that Definition 3 assumes that every repeated execution of a task by the same resource, i.e., for the same case, has the same duration. In practice, this needs not to be the case, i.e., the 2nd execution of the same activity by the same resource may be generally significantly faster or slower. Clearly, the signature of the domain of the $\hat{\Delta}$-function can be extended to cover more contextual attributes than just activity and resource, e.g., other available attributes. However, notably, it is likely that the most effective context to use depends on both the process under study as well as the corresponding event data.

4.3 Feasibility Checking

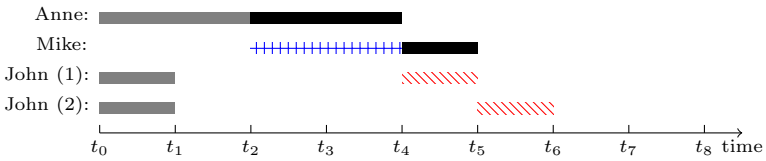
As a second step of our proposed framework, we assess the feasibility of the schedule. The feasibility check consists of two major steps, i.e., *task duration replacement* and *constraint-compliance checking*. The latter step depends primarily on either the constraints used to generate the schedule (in case scheduling is applied) or on domain knowledge regarding general scheduling feasibility, e.g., task may only be executed during work hours. However, the former step, i.e., *task duration replacement*, can be performed either in a *conservative* or in a *progressive manner*.

Given some schedule $S\in\mathcal{S}$ and $\tau=(a,r,t_1,t_2)$. In the conservative replacement strategy, we only replace task durations that are overly optimistic in the given schedule, i.e., if $\Delta(\tau)<\hat{\Delta}(a,r)$ we replace τ in S by $\tau'=(a,r,t_1,t_1+\hat{\Delta}(a,r))$. However, if $\Delta(\tau)>\hat{\Delta}(a,r)$, we keep τ in the schedule, i.e., as it is assumed to take longer than the estimator and thus, is not expected to lead to problems. Reconsider Fig. 1b, and assume that we predict *Anne* to require 4 time units for "Invite Reviewers", *Mike* to require 3 time units for "Collect Reviews", and *John* only 1 time unit for the "Decide" activity. Fig. 3a depicts the revised schedule, using the conservative strategy. Observe that the schedule inviting reviewers is expected to overlap with the review collection. Similarly, the decision activity is expected to partially overlap with the review collection.

In the progressive replacement strategy, we replace τ in S by $\tau'=(a,r,t_1,t_1+\hat{\Delta}(a,r))$ if $\Delta(\tau)\neq\hat{\Delta}(a,r)$. Hence, in the progressive strategy, we allow tasks to be shortened in the expected duration as well. Observe that the consequences of shortening tasks in the schedule are more profound. Reconsider the previous example and consider Fig. 3b, illustrating the corresponding result of applying the progressive strategy. The "Decide" task of *John* is expected to only take one time-unit. However, as the original schedule reserved the time-frame $[t_4,t_6]$, the activity can be positioned to be starting anywhere in the time-frame $[t_4,t_5]$.



(a) Example of *conservative task duration replacement*. The tasks of *Anne* and *Mike* are expected to take longer (extended time-frame highlighted in black). John is expected to take only one time-unit for the "Decide" activity. However, the conservative strategy does not alter the scheduled "Decide" task for John.



(b) Example of *progressive task duration replacement*. The progressive strategy shortens the scheduled "Decide" task for John. We are able to schedule the start of the "Decide" activity for *John* in-between t_4 and t_5. The earliest and latest scheduling of the decide activity are both visualized (*John (1)* and *John (2)*) respectively.

Fig. 3: Different examples of task duration replacement of the schedule depicted in Fig. 1b, i.e., covering both the *conservative* (Fig. 3a) and *progressive strategy* (Fig. 3b).

5 Evaluation

In this section, we evaluate our proposed event-data-driven schedule feasibility check. We present the experimental setup in Section 5.1. Section 5.2 presents the results.

5.1 Experimental Setup

In this section, we present the general experimental setup of our evaluation. We present the research questions considered, discuss the event data used and the general setup of the experiments, including the preprocessing steps conducted to prepare the event data for our experiments.[3]

Research Questions The primary focus of our evaluation is an assessment of the computational aspects. In the context of our evaluation, we aim to answer the following research questions:

1. Is there a (significant) difference in the parameter sensitivity of the proposed estimators?

[3] The source code of our experiments is publicly available at https://github.com/HannesHf/FeasibilityCheckingofProcessSchedules.

Table 2: Descriptive statistics of the BPIC17 event log, used in the experiments.

Events	1,202,267
Events with Start	128,227
Events with Complete	475,306
Resulting Tasks	44,503
Total Timeframe	Jan. 2016 - Jan. 2017
Training Window	Jan. 2016 - Apr. 2016
Test Data (Schedule Proxies)	May 2016 - Oct. 2016

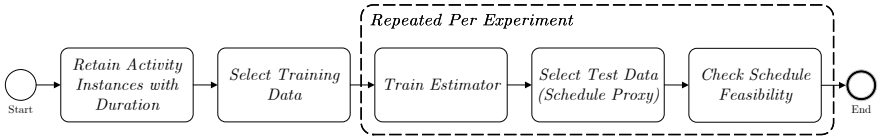


Fig. 4: Schematic control-flow view of the steps conducted in the experiments.

2. Is there a (significant) difference in the computational complexity of the proposed estimators?
3. Which distributions are most often fitted when using the distribution-based estimator?

Event Data Selection In our experiments, we use *real event data*. A primary requirement of the event data is the availability of both start and end timestamps as well as resource information. However, most publicly available event data (https://data.4tu.nl/) only records one timestamp per event. In the BPI Challenge 2017 log (BPIC17) [7] (https://data.4tu.nl/articles/dataset/BPI_Challenge_2017/12696884), a subset of the recorded events describe both a start and end timestamp (respectively $128,227$ start and $475,306$ end timestamps, out of a total of $1,202,267$ events). Additionally, the event data is mapping of a task to a specific resource, where the set resource is to some extent constant. The event log pertains to a loan application process of a Dutch financial institute, and as such several tasks performed by human resources differ in duration. We match the events describing a start and complete timestamp to create tasks. Note that only a small subset of the events can be matched because either the started task is aborted, i.e., there exists no corresponding complete timestamp, or the events are atomic events, i.e., there is no corresponding start timestamp. Consider Table 2, in which we present general descriptive statistics of the BPIC17 event log in the context of our experiments.

Setup Fig. 4 depicts the control flow of the experimental setup. In the first step, we retain all activity instances with a non-zero duration. The final number of extracted tasks for our experiments is $44,503$. We use the event data both as

a training set, i.e., to compute the task estimator, as well as to present schedules (i.e., *schedule proxies*). To create a training data set we use the tasks occurring in the time period *Jan. 2016 - Apr. 2016*, yielding 13,131 tasks. We use the events recorded in the time period *May 2016 - Oct. 2016* as *proxy schedules*. The sizes of the derived schedules are (# tasks): 05/16: 3,383; 06/16: 4,408; 07/16: 4,289; 08/16: 3,648; 09/16: 4115; 10/16: 3,715 We use all activity instances between Jan. 1st, 2016, and Apr. 30th, 2016, as a training window (cf. Table 2). For every experiment, we repeat the subsequent three steps, i.e., *training the estimator*, *selecting the test data*, and finally *checking the schedule feasibility*.

We consider the empirical average estimator ($\hat{\Delta}_{\texttt{avg},k}$), with $k \in \{-2, -1.5, \ldots, 1.5, 2\}$. For the distribution based estimator ($\hat{\Delta}_{\texttt{dist},\hat{f},k}(a,r)$), we perform distribution fitting on the *normal*, *exponential*, *beta* and *gamma* distributions and use $k \in \{0.01, 0.05, 0.1, 0.2, 0.3, \ldots, 0.8, 0.9, 0.95, 0.99\}$ for feasibility checking. As no schedules are available for BPIC17, we select a subset of the event log to represent a schedule, i.e., as test data (which we refer to as a *proxy schedule*). In terms of task replacement, we only apply the conservative replacement strategy.

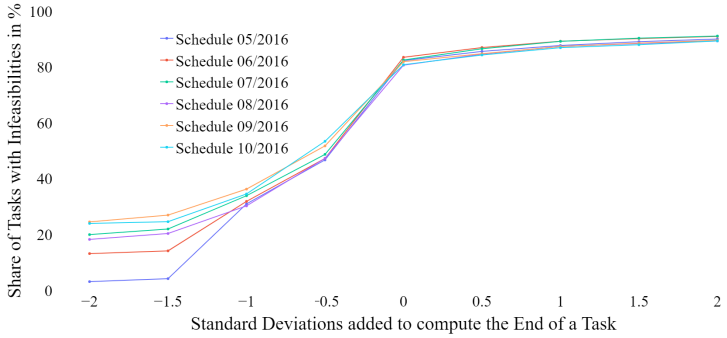
5.2 Results

In this section, we discuss the results of our experiments. We first consider *parameter sensitivity*, after which we focus on *computational complexity*. Finally, we briefly investigate the *most frequently occurring distribution* for the task-duration estimator.

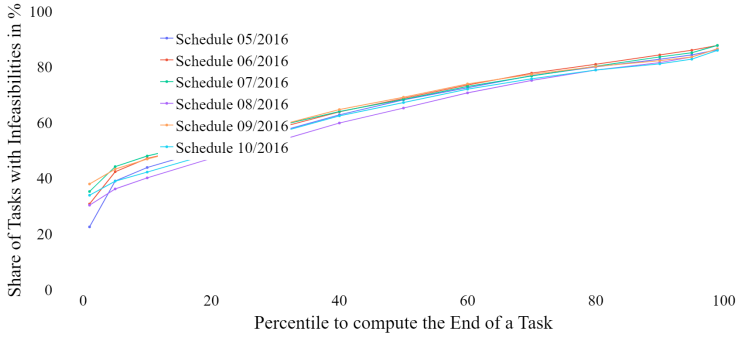
Parameter Sensitivity In this section, we compare the parameter sensitivity of the average-based estimator against the distribution-based estimator. In particular, we investigate the sensitivity of both approaches in terms of their conflict-detection ratio for different parameterizations of the algorithm. Generally, we compute a revised schedule, for which we measure the ratio of the number of tasks that overlap with another task to be executed by the same resource.[4] Let z denote the number of tasks that have such an issue; the ratio q we compute is $q = \frac{z}{|S'|}$. Hence, if q=0, no problems have been detected. If q=1, all tasks have a problem.

Consider Fig. 5, in which we depict the results for both estimators. Observe that, in the figures, the previously described q values are listed as percentages, and, similarly, for the distribution-based estimator (Fig. 5b), the k-values are represented as percentages. The empirical-average-based estimator (Fig. 5a) shows a much stronger parameter sensitivity compared to the distribution-based estimator (Fig. 5b). This makes sense as, for example, for a normal distribution, a value of k close to 0 or 1 for the distribution-based approach describes the results at $\overline{x} +/- 2\sigma_{\overline{x}}$ for the empirical-average-based approach. In that regard, even though different underlying distributions are used for activity-resource combinations, using $k \in \{0.2 \ldots 0.8\}$ for the distribution-based estimator broadly covers $\overline{x} +/- \sigma_{\overline{x}}$

[4] Observe that, since we use real event data, we are not aware of the actual constraints of the process.



(a) Measurement of schedule compliance/feasibility based on mean values and varied standard deviations



(b) Measurement of schedule compliance/feasibility based on distribution fitting with varying percentiles

Fig. 5: Results for parameters sensitivity of the two proposed estimators, i.e., *empirical average* (Fig. 5a) and *distribution-based* (Fig. 5b)

for the empirical-average-based. Generally, the distribution-based approach is less sensitive to parameter changes, except for the extremes of parameter k.

Interestingly, in both cases, the highly "optimistic" values of the parameters (i.e., -2 for Fig. 5a and close to 0 for Fig. 5b) still yield a detection value of q=0.2. Upon inspection of the results, we observed that this is because some activity-resource pairs infrequently occur in the training data, even yielding large estimated values for optimistic parameter settings. In other cases, some tasks still overlap, i.e., even though the estimated duration is very low. In such cases, either the tasks were already overlapping in time in the real execution (i.e., used as a proxy), or the low duration is still an overestimate of the actual duration at that point in time.

Computational Complexity Here, we focus on the computational complexity of the estimators. Clearly, computing the average and standard deviation of a set

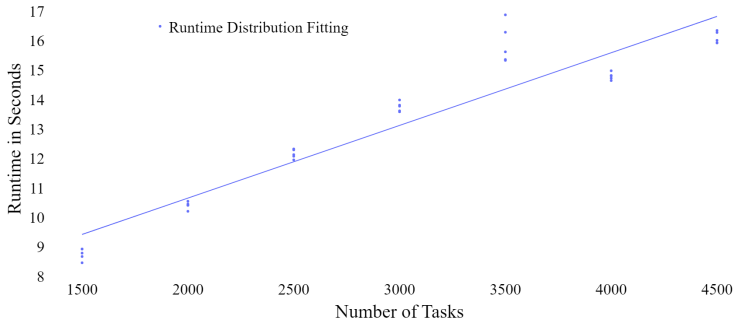


Fig. 6: Runtime of distribution fitting and computation of percentiles per size of event log combined with a linear trend-line (R^2=0.895132). The experimental results show a linear trend in terms of the number of activity instances present in the training data.

Table 3: Distribution fitting results.

Distribution	***Absolute***	***Relative***
Beta	99	54.4%
Gamma	60	33%
Exponential	22	12.1%
Normal	1	0.5%

of numbers is a linear operation in terms of its input. Even for extremely large numbers, such statistic can be computed on commodity hardware, as such, we refrain from presenting run-time measurements. Similarly, schedule revision has a theoretical run-time of $O(|S|)$. Measuring the number of errors has a theoretical run-time of $O(|S|^2)$. As we expect schedules to be of reasonable size, we do not further investigate the runtime of said check.

Distribution fitting may have a wider range of theoretical runtime complexity and is, as such, more interesting to study, i.e., particularly as we perform multiple tests. In Fig. 6, we present the run-time of the training time of the distribution-based estimator. To generate training data of increasing sizes, we randomly sampled relevant activity instances from the BPIC17 data (cf. Table 2). We apply distribution fitting on top of the collection of recorded instances. We observe that the results show a linear trend regarding the number of instances.

Frequently Occurring Distributions Here, we focus on the most often occurring distributions, i.e., when using the distribution-based estimator. Consider 3, in which we depict the number of times a given distribution was deemed the best fit for an activity-resource combination. Due to its relatively flexible nature, the beta distribution is most often used, i.e., in approximately $54,4\%$. In roughly 33%, the gamma distribution yields the best fit, and in 12.1%, the exponential

distribution. Logically, the normal distribution rarely fits well, i.e., only in one case it returned the best fit. However, in some instances, limited training data may be available for the distribution fitting.

6 Conclusion

The application of scheduling often overlooks the availability of event data and does not recognize the vast potential such data can offer for an increased overall quality of generated schedules. At the same time, process-oriented research fields exploiting event data, i.e., process mining, tend to ignore the existence and importance of scheduling the to-be-executed process activities. In this paper, we proposed a novel event-data-driven schedule feasibility-checking framework to bridge the abovementioned gap. Our framework proposes to learn duration estimators based on event data, which are subsequently used to revise the duration of task allocations in a given schedule. Our initial results confirm the general applicability of our proposed framework and show that the proposed framework instantiations are computationally feasible.

As future work, we plan to conduct several case studies, enabling the use of more realistic schedules and corresponding event data. Secondly, we also aim to allow the replacement of scheduled tasks by shorter expected activity duration. We additionally aim to work with *buffers*, i.e., portions of idle time in which the scheduled activities are allowed to exceed their initially scheduled time.

References

1. van der Aalst, W.M.P.: Process Mining - Data Science in Action, Second Edition. Springer (2016)
2. Arias, M., Munoz-Gama, J., Sepúlveda, M., Miranda, J.C.: Human resource allocation or recommendation based on multi-factor criteria in on-demand and batch scenarios. European Journal of Industrial Engineering **12**(3), 364–404 (2018)
3. Bala, S., Cabanillas, C., Mendling, J., Rogge-Solti, A., Polleres, A.: Mining project-oriented business processes. In: Motahari-Nezhad, H.R., Recker, J., Weidlich, M. (eds.) Business Process Management - 13th International Conference, BPM 2015, Innsbruck, Austria, August 31 - September 3, 2015, Proceedings. Lecture Notes in Computer Science, vol. 9253, pp. 425–440. Springer (2015)
4. Bellaaj, F., Sellami, M., Bhiri, S., Maamar, Z.: Obstacle-aware resource allocation in business processes. In: Abramowicz, W. (ed.) Business Information Systems. pp. 207–219. Springer International Publishing, Cham (2017)
5. Berkenstadt, G., Gal, A., Senderovich, A., Shraga, R., Weidlich, M.: Queueing inference for process performance analysis with missing life-cycle data. In: van Dongen, B.F., Montali, M., Wynn, M.T. (eds.) 2nd International Conference on Process Mining, ICPM 2020, Padua, Italy, October 4-9, 2020. pp. 57–64. IEEE (2020)
6. de Man, J.C., Strandhagen, J.O.: Spreadsheet application still dominates enterprise resource planning and advanced planning systems. IFAC-PapersOnLine **51**(11), 1224–1229 (2018), 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018

7. van Dongen, B.F.: BPI Challenge 2017 (2 2017). https://doi.org/10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b, https://data.4tu.nl/articles/dataset/BPI_Challenge_2017/12696884
8. van Eck, M.L., Firat, M., Nuijten, W.P.M., Sidorova, N., van der Aalst, W.M.P.: Human performance-aware scheduling and routing of a multi-skilled workforce. Complex Syst. Informatics Model. Q. **12**, 1–21 (2017)
9. Havur, G., Cabanillas, C., Mendling, J., Polleres, A.: Resource allocation with dependencies in business process management systems. In: Rosa, M.L., Loos, P., Pastor, O. (eds.) Business Process Management Forum - BPM Forum 2016, Rio de Janeiro, Brazil, September 18-22, 2016, Proceedings. Lecture Notes in Business Information Processing, vol. 260, pp. 3–19. Springer (2016)
10. Havur, G., Cabanillas, C., Polleres, A.: Benchmarking answer set programming systems for resource allocation in business processes. Expert Syst. Appl. **205**, 117599 (2022)
11. 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**(11), 2405–2429 (nov 2022)
12. Kinast, A., Doerner, K.F., Rinderle-Ma, S.: Combining metaheuristics and process mining: Improving cobot placement in a combined cobot assignment and job shop scheduling problem. Procedia Computer Science **200**, 1836–1845 (2022), 3rd International Conference on Industry 4.0 and Smart Manufacturing
13. Leung, J.Y. (ed.): Handbook of Scheduling - Algorithms, Models, and Performance Analysis. Chapman and Hall/CRC (2004)
14. Loeve, M.: Probability theory: Third edition. Dover Publications, Mineola, NY, 3 edn. (2017)
15. Low, W.Z., van der Aalst, W.M.P., ter Hofstede, A.H.M., Wynn, M.T., Weerdt, J.D.: Change visualisation: Analysing the resource and timing differences between two event logs. Inf. Syst. **65**, 106–123 (2017)
16. Mans, R., Russell, N.C., van der Aalst, W.M.P., Moleman, A.J., Bakker, P.J.M.: Schedule-aware workflow management systems. Trans. Petri Nets Other Model. Concurr. **4**, 121–143 (2010)
17. Mans, R.: Workflow support for the healthcare domain. Ph.D. thesis, Industrial Engineering and Innovation Sciences (2011)
18. Ouelhadj, D., Petrovic, S.: A survey of dynamic scheduling in manufacturing systems. J. Sched. **12**(4), 417–431 (2009)
19. Patel, J.K., B., C.: Handbook of the normal distribution, second edition. CRC Press, Boca Raton, FL, 2 edn. (1996)
20. Pinedo, M.L.: Scheduling - Theory, Algorithms, and Systems, Fifth Edition. Springer (2016)
21. Senderovich, A., Booth, K.E.C., Beck, J.C.: Learning scheduling models from event data. In: Benton, J., Lipovetzky, N., Onaindia, E., Smith, D.E., Srivastava, S. (eds.) ICAPS 2018, Berkeley, CA, USA, July 11-15, 2019. pp. 401–409. AAAI Press (2019)
22. Senderovich, A., Weidlich, M., Yedidsion, L., Gal, A., Mandelbaum, A., Kadish, S., Bunnell, C.A.: Conformance checking and performance improvement in scheduled processes: A queueing-network perspective. Inf. Syst. **62**, 185–206 (2016)
23. Taha, H.A.: Operations research: An introduction. Pearson Education, Harlow, England and London and New York and Boston and Amsterdam and Munich, tenth edition, global edition edn. (2017)
24. Winston, W.L., Goldberg, J.B.: Operations research: applications and algorithms, fourth edition. Thomson Brooks/Cole Belmont (2004)