IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 23 March 2023, accepted 13 April 2023, date of publication 17 April 2023, date of current version 21 April 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3267538

RESEARCH ARTICLE

Optimizing Resource Allocation Based on Predictive Process Monitoring

GYUNAM PARK[1,2] AND MINSEOK SONG[1], (Member, IEEE)
[1]Department of Industrial and Management Engineering, Pohang University of Science and Technology (POSTECH), Nam-gu, Pohang 37673, South Korea
[2]Process and Data Science Group (PADS), Department of Computer Science, RWTH Aachen University, 52074 Aachen, Germany

Corresponding author: Minseok Song (mssong@postech.ac.kr)

This work was supported in part by the National Research Foundation of Korea (NRF) funded by the Korean Government [Ministry of Science and ICT (MSIT)] under Grant 2021R1A2C3012292; and in part by the Pohang Iron & Steel Company (POSCO), South Korea, under the Institute of Process Engineering (IPE) through the Joint Research on Smart Solution (JRSS) and Steel Science Project.

ABSTRACT Recent breakthroughs in predictive business process monitoring equip process analysts with predictions on running process instances, supporting the elicitation of proactive measures to mitigate risks that can be caused by the process instance. However, contrary to active research on providing various predictions and improving the accuracy of prediction models, the practical application of such predictions has been left to the subjective judgment of domain experts. In this work, we explore the exploitation of the insights from predictive information for the actual process improvement in practice. Concretely, we focus on improving resource allocation in business processes where the goal is to allocate appropriate resources to tasks at the proper time. Based on design science methodology, we develop a two-phase method to improve resource allocation by leveraging predictions. Based on the method, we instantiate an algorithm to optimize *total-weighted completion time* and evaluate its effectiveness and efficiency. From an academic standpoint, our work demonstrates the combination of predictions using machine learning and optimizations based on scheduling. From a practical standpoint, our work provides a general approach to optimize resource allocations for different objectives using predictions.

INDEX TERMS Resource allocation, online operational support, process improvement, Bayesian neural network, minimum cost and maximum flow algorithm.

I. INTRODUCTION

Due to the increasing importance of business process improvement, organizations strive to analyze the ongoing instances in their processes to which they can apply immediate management actions when problems arise [1]. In this regard, predictive business process monitoring has obtained considerable attention. It aims to improve business processes by providing timely information (e.g., remaining time, risk probability, performance indicators, and next event of a running instance), which enables proactive and corrective actions [2]. Various approaches have been developed to predict such predictive information based on annotated transition systems [2], machine learning algorithms [3], and statistics [4].

The associate editor coordinating the review of this manuscript and approving it for publication was Md. Abdur Razzaque.

However, despite the recent development in the field of predictive business process monitoring, an important question still remains unanswered: *How can we achieve actual process improvement using such timely information?* Most of the existing studies focus on constructing more accurate predictive models while leaving the concrete actions up to the subjective judgment of domain experts [5]. Indeed, it is essential to escalate the insightful predictions to the realm of prescriptive actions [6] so that the real values can be brought to the process owners.

In this work, we focus on turning such insights into actions to optimize resource allocation. Resource allocation in business processes refers to the allocation of appropriate resources for tasks of process instances at the right time. Efficient resource allocation has been recognized as an important issue in business process management owing to its ability to improve business processes by fostering

productivity, balancing resource usage, and reducing execution costs [7], [8].

Existing techniques for resource allocation require information such as the release time, service time, and sequence of required tasks of a process instance [9]. However, in real-life business processes, we often have limited information on process instances (so-called *non-clairvoyant* resource allocation problems). In such problem settings, the existing techniques cannot be directly applied due to the missing information [10].

We address this problem using a design science methodology [11], [12].

1) **Design as an artifact**: We develop a method supporting resource allocation in *non-clairvoyant* settings by leveraging predictions. The approach consists of two phases. The first phase is to construct models that predict scheduling parameter values. The second phase is to predict the values of parameters using the prediction models and compute the optimal resource allocation.
2) **Problem relevance**: The practical relevance of the problem lies with the increasing importance of business process improvement and the call for leveraging predictive insights for process improvement [1].
3) **Design as a search process**: The existing literature on predictive business process monitoring and resource allocation inspires the design search process from which we draw ideas to design the approach and evaluate its effectiveness and efficiency.
4) **Research rigor**: We ensure the rigor of the design process by mathematically formalizing all components of the proposed approach.
5) **Design evaluation**: We instantiate the method to optimize *Total-Weighted Completion Time (TWCT)* of *weight-aware Non-clairvoyant Process Instances (NoPIs)*. Moreover, we evaluate the scheduling performance of the instantiated approach in a simulated process and a real-life process of the emergency department in a tertiary hospital by comparing it with a baseline approach. We also analyze the effects of the prediction accuracy on the scheduling performance and empirically inspect the minimum level of the prediction accuracy to achieve acceptable resource allocations.
6) **Research contributions**: First, we develop a two-phase method to optimize *non-clairvoyant* resource allocation by leveraging predictive information. Second, we have evaluated the scheduling performance of the instantiation using both simulated and real-life business processes. Third, we have analyzed the minimum level of prediction accuracy to achieve reasonable resource allocations.

The remainder of this paper is organized as follows. section II presents the related works. section III provides preliminaries on event data, resource allocation problems, and non-clairvoyant resource allocation problems. Besides, section IV introduces the suggested approach. In section V, a specific type of non-clairvoyant resource allocation problem, *optimizing TWCT of weight-aware NoSIs*, is explained. section VI presents an instantiated method based on the approach to solve the problem and section VII evaluates the method. Finally, section IX concludes the paper.

II. RELATED WORK

This section introduces relevant techniques for predictive business process monitoring. Afterward, we explain the existing work for connecting predictions to process improvement and optimizing resource allocation in business processes and discuss its limitations.

A. PREDICTIONS IN BUSINESS PROCESSES

Predictive business process monitoring has been actively studied [3]. A plethora of techniques has been suggested to predict various process-related values such as time, performance indicators, next event, and risk factors.

Earlier approaches deployed process models, which are automatically discovered from event data, and annotated them with historical information by replaying process instances. The annotated historical information of the process model is used for learning prediction models. For instance, van der Aalst et al. [4] suggested a framework for the prediction of time-related properties by deploying transition systems and annotating each state with measurement values. Folino et al. [13] extended this framework by clustering process instances using context features. It has been further enhanced by adding SVR and Naïve Bayes classifiers to the additional attributes of events [2].

More recent techniques are model-unaware, deploying more complex machine learning models. In particular, recent breakthroughs in deep neural networks have enabled developments of predictive models showing higher accuracy than traditional machine learning models. For instance, Evermann et al. [14] proposed a method based on LSTM neural networks by encoding features based on embedding, showing the state-of-the-art accuracy of the prediction. Besides, Tax et al. [15] suggested an LSTM-based model by encoding features based on one-hot-encoding and using multitask learning. Mehdiyev et al. [16] extends neural network-based approaches by providing an architecture composed of unsupervised stacked autoencoders and supervised fine-tuning with n-gram features.

The explainability of predictive models has been more relevant due to the nature of neural network-based predictive models that the learned structure is difficult to intuitively represent, known as the black-box problem [17]. In fact, practitioners are reluctant to adopt predictive models providing no explanations regarding the resulting prediction. Stierle et al. [18] deployed Graph-based Neural Networks (GNNs) to infer temporal dependencies existing in event data to determine relevance scores of process activities in terms of a performance measure. Hans et al. [19] proposed a technique to estimate uncertainties of remaining time predictions using Bayesian Neural Networks (BNNs).

B. IMPROVEMENTS IN BUSINESS PROCESSES

Contrary to extensive research on predictive business process monitoring, few studies have provided guidance and recommendations using predictions. Conforti et al. [20] propose a recommendation system to optimize resource allocation in business processes based on the risk predictions of running instances. In [21], a prescriptive alarm system generates alarms when process instances show problematic behaviors. A recommendation system [22] suggests the next best activity by analyzing the next most likely activities in terms of key performance indicators.

Some studies have been conducted to analyze the effects of improvement actions. In [23], a technique is proposed to analyze the impacts of improvement actions in terms of the structure, operation, and performance of business processes. Dees et al. [24] demonstrate the importance of making efficient decisions on actions based on accurate predictions, reporting that premature actions may lead to undesirable results. In [25], the authors report the importance of objective, unbiased actions for improving key performance indicators, calling for more systematic approaches to support the decisions on effective actions.

Recently, several approaches have been proposed to turn insights from process analytics into management actions, e.g., sending alerts to employees, changing configurations in information systems, and handling problematic process instances, in the field of action-oriented process mining. A commercial process mining tool, *Celonis Action Engine*, analyzes event data, generates signals based on the analysis, and executes predefined actions to the target information system according to the signals [26]. A general framework for action-oriented process mining is suggested in [27], and the framework is instantiated using digital-twin interface models in [28].

The authors of this work have proposed a method to optimize a resource allocation problem using LSTM networks and the minimum cost and maximum flow network algorithm [29]. The findings from the previous study work as foundations for developing the proposed framework.

C. RESOURCE ALLOCATION IN BUSINESS PROCESSES

Finding optimal resource allocations in business processes shares some commonalities with Job Shop Scheduling Problem (JSSP) [20]. A JSSP deals with a set of resources performing operations on jobs (i.e., process instances), where each job has predefined tasks (i.e., activities) through the resources. The goal is to find sequences of jobs on resources while optimizing an objective, e.g., minimizing total completion time. It is an NP-hard problem, considered to be among the most intractable problems [30].

A large body of literature exists that deals with JSSPs [31], including approaches such as shifting bottleneck heuristic and local search [32]. Despite their good approximation of optimality, it has been reported that applying them requires a huge computational cost, particularly when the problem size is large [33]. The high computational cost limits practical applications of such techniques in real-life business processes due to the dynamic nature of the process [34].

Instead, dispatching rules have been widely employed in the industry owing to their high computational efficiency and robustness to uncertainty [9]. However, existing studies make simplified assumptions in the definition of the problems, thus hindering their application in the context of business process management. For instance, the sequence of activities is assumed to be invariant among the different scheduling instances, whereas each activity has its unique single resource. Moreover, some of them assume one or two resources that can perform any activities in business processes. Instead, in real-life business processes, we have different types of scheduling instances, which have various sequences of activities, and the resources have their own availability to perform some of the activities in the process.

There have been efforts to solve non-clairvoyant scheduling problems with the tight upper and lower bounds of its scheduling performances. For example, Becchetti et al. proposed a novel method for the resource allocation of non-clairvoyant scheduling instances in terms of average flow time minimization in parallel machine environments in [35]. IM et al. [36] have suggested algorithms to optimize the schedule of the scheduling instance that have different arrival times and sizes, whereas resources have arbitrary packing constraints in a non-clairvoyant environment.

III. PRELIMINARIES

This section introduces event data and resource allocation problems.

A. EVENT DATA

An event is characterized by its attributes such as case identifier, activity, resource, and timestamp.

Definition 1 (Event): Let $\mathbb{U}_{event}$ be the universe of events, $\mathbb{U}_{case}$ the universe of case identifiers, $\mathbb{U}_{act}$ the universe of activity names, $\mathbb{U}_{res}$ the universe of resource names, and $\mathbb{U}_{time}$ the universe of timestamps. Let AN be the set of attribute names such that $\{case, act, res, time\} \subseteq AN$. For $e \in \mathbb{U}_{event}$ and $an \in AN$, $\pi_{an}(e)$ is the value of attribute an for event e. If event e does not have the attribute, we denote $\pi_{an}(e) = \perp$.

For instance, given event $e \in \mathbb{U}_{event}$, $\pi_{case}(e)$, $\pi_{act}(e)$, $\pi_{res}(e)$, and $\pi_{time}(e)$ indicate the case identifier, activity, resource, and timestamp of the event, respectively.

Definition 2 (Trace): A trace $\sigma \in \mathbb{U}^*_{event}$ is a sequence of different events: $\sigma = \langle e_1, e_2, \ldots, e_n \rangle$ such that, for $1 \leq i < j \leq |\sigma|$, $e_i \neq e_j$. $\mathbb{U}_{trace}$ is the set of all possible traces.

Similar to events, cases also exhibit mandatory attributes such as a trace. An event log is a set of cases.

Definition 3 (Case, Event Log): Let AN be the set of attribute names such that $\{trace\} \in AN$. For $c \in \mathbb{U}_{case}$ and $an \in AN$, $\pi_{an}(c)$ is the value of attribute an for case c. If case c does not have the attribute, we denote $\pi_{an}(c) = \perp$. An event log $EL \subseteq \mathbb{U}_{case}$ is the set of cases. $\mathbb{U}_{EL}$ is the set of all possible event logs.

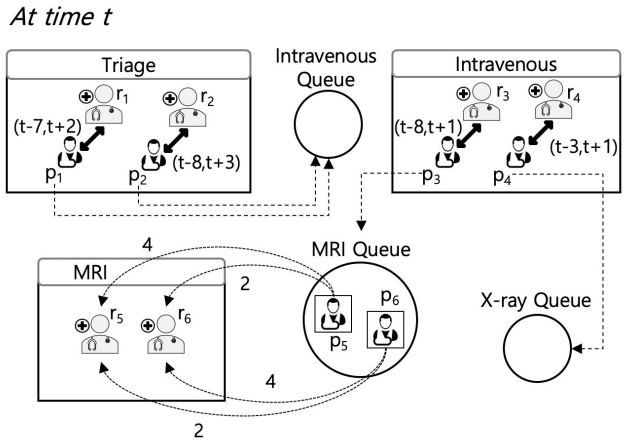


FIGURE 1. Snapshot of the business process in a hospital at time t. We will use the situation described in this figure as the basis for examples in the remaining of the paper.

B. RESOURCE ALLOCATION PROBLEM

Figure 1 presents a snapshot of a part of a healthcare process at time t. The rectangles indicate the different activities, whereas the circles indicate the queues for the treatments where patients (i.e., cases) need to wait before medical staff (i.e., resources) are assigned. There are six patients (i.e., $p_1, \ldots, p_6$) and six medical staff (i.e., $r_1 \ldots, r_6$), and the bidirectional arcs between them indicate that the patient is being operated by the medical staff, with the annotated tuple representing the start and completion times of the operation. For instance, p_1 is operated by r_1 from $t-7$ to $t+2$, which denotes that the service time is 9 time units. Moreover, the dotted arc means the next treatment of the patient. For instance, the next operation of p_3 is the MRI, i.e., p_3 will enter the queue for the MRI at $t+1$ when the current activity is completed. Furthermore, two patients are waiting to undergo MRI inside the queue. The curved arc indicates the service time required by each resource to complete the MRI of the patient.

Resource allocation problem is to find the appropriate *work assignments* between activities of *process instances* (e.g., patients) and resources (e.g., medical staff) at the right time. A *process instance*, also called *scheduling instance*, involves *scheduling parameters*, e.g., the required activity and its service time, utilized for the allocation of appropriate resources in terms of an *objective* of an organization, e.g., minimizing makespans. The different techniques for resource allocation may require different parameters. For instance, Figure 1 depicts the parameters of p_1 such as the current activity (i.e., *triage*), the service time by r_1 (i.e., 9 time units), and the next activity (i.e., *intravenous*).

Most of the techniques for resource allocation require *a base set of parameters* involving 1) complete sequences of activities (i.e., tasks) that need to be performed and 2) the corresponding service times by available resources [37]. We use the following notations in the remainder of this paper:

- $\mathbb{U}_{pm}$ is the universe of scheduling parameters.
- ml denotes the max. length of the activity sequence of cases in a business process.
- $a_i \in \mathbb{U}_{pm}$ indicates the *i-th activity* of a scheduling instance, where $i < ml$.
- $pt_{i,r} \in \mathbb{U}_{pm}$ denotes the service time of the *i-th activity*, i.e., a_i, when operated by resource $r \in \mathbb{U}_{res}$.
- AS is the set of parameters associated with the sequence of activities, i.e., $AS = \{a_i \in \mathbb{U}_{pm} | 0 < i < ml\}$.
- PT denotes the set of parameters associated with the service times of activities by the different resources: $PT = \{pt_{i,r} \in \mathbb{U}_{pm} | 0 < i < ml \wedge r \in \mathbb{U}_{res}\}$.
- BP is the base set of parameters, i.e., $BP = AS \cup PT$.

A scheduling instance consists of the case identifier associated with the one in event logs, and the parameter mapping that assigns values to its associated parameters.

Definition 4 (Scheduling Instance): A scheduling instance $si = (c, pmap) \in \mathbb{U}_{case} \times (\mathbb{U}_{pm} \nrightarrow \mathbb{U}_{val})$ is a pair of a case identifier c and a parameter mapping $pmap$. Given $si = (c, pmap)$, we indicate $\pi_{case}(si) = c$ and $\pi_{pmap}(si) = pmap$. $\mathbb{U}_{si}$ is the set of all possible scheduling instances.

Assume that the sequence of activities planned for $p_1 \in \mathbb{U}_{case}$ in Figure 1 includes *triage*, *intravenous*, *MRI*, and *evaluation* in order. The patient is represented as $(p_1, pmap_1)$, where $pmap_1(a_1) = triage$, $pmap_1(a_2) = intravenous$, $pmap_1(a_3) = MRI$, $pmap_1(a_4) = evaluation$, $pmap_1(a_5) = \perp$, $pmap(pt_{1,r1}) = 9$, etc.

In real-life business processes, missing values of scheduling parameters is common in scheduling instances, limiting the application of established techniques for resource allocation [10]. We call such instances *Non-clairvoyant Scheduling Instance (NoSI)*.

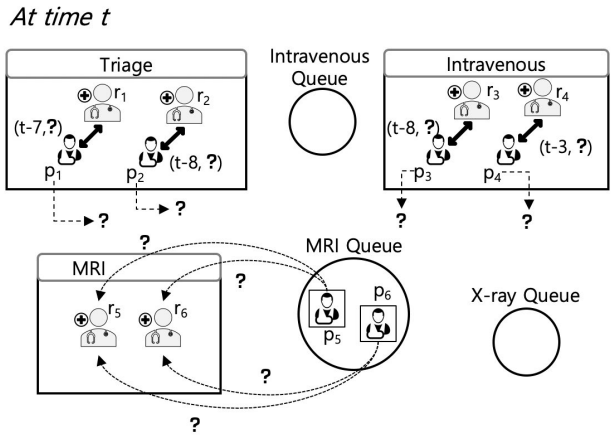


FIGURE 2. Snapshot of the business process in a hospital at time t where the patients are non-clairvoyant scheduling instances.

Figure 2 describes the same situation as Figure 1, except the scheduling instances being NoSIs. We may not know the next activity of patients before the completion of the current operation. For instance, we do not know the next activity of p_1 as the patient is currently performing *triage* activity.

The next activity is known right after p_1 completes *triage* activity. Moreover, we have no information on the expected service time required by different resources to operate on the patients. For instance, we have no information on how much time it will take for r_5 and r_6 to operate on p_5. p_1 is represented by $si_1 = (p_1, pmap_1)$, where $pmap_1(a_1) = triage$, $pmap_1(a_2) = \perp$, $pmap_1(a_3) = \perp$, $pmap(pt_{1,r1}) = \perp$, etc.

Definition 5 (Work Assignment): Let $|c|$ denote the number of activities that need to be operated for case $c \in \mathbb{U}_{case}$. A work assignment $wa = (c, i, r, t_s, t_c) \in \mathbb{U}_{case} \times \mathbb{N} \times \mathbb{U}_{res} \times \mathbb{U}_{time} \times \mathbb{U}_{time}$ is a tuple of a case, an index of the current activity, a resource, a start timestamp, and a complete timestamp such that $i \leq |c|$ and $t_s \leq t_c$. Given $wa = (c, i, r, t_s, t_c)$, we indicate $\pi_{case}(wa) = c$, $\pi_i(wa) = i$, $\pi_{res}(wa) = r$, $\pi_{ts}(wa) = t_s$, and $\pi_{tc}(wa) = t_c$. We let $\mathbb{U}_{wa}$ denote the set of all possible work assignments.

For instance, the work assignment relevant to the *triage* activity of p_1 in Figure 1 is represented as $wa' = (p_1, 1, r_1, t-7, t+2)$, i.e., the first activity of p_1 is performed by r_1 from $t-7$ to $t+2$.

A schedule contains the work assignments of scheduling instances. The work assignments of the schedule should satisfy the following constraints. First, it should cover all the activities planned for the scheduling instances. In addition, each activity in the sequence is performed only once, and only one operation is processed at a given time. Furthermore, each resource can perform only one activity at the same time. We formally define the constraint as follows.

Definition 6 (Schedule): A schedule $S \subseteq \mathbb{U}_{wa}$ is a collection of work assignments such that

- $\forall_{\{wa,wa'\} \subseteq S \wedge \pi_{case}(wa) = \pi_{case}(wa')} \ \pi_i(wa) \neq \pi_i(wa')$,
- $\forall_{c' \in \{\pi_{case}(wa) | wa \in S\}} \ |c'| = |\{wa \in S | \pi_{case}(wa) = c'\}|$,
- $\forall_{\{wa,wa'\} \subseteq S \wedge \pi_{res}(wa) = \pi_{res}(wa')} \ \pi_{ts}(wa) \geq \pi_{tc}(wa') \vee \pi_{ts}(wa') \geq \pi_{tc}(wa)$, and
- $\forall_{\{wa,wa'\} \subseteq S \wedge \pi_{case}(wa) = \pi_{case}(wa')} \ \pi_{ts}(wa) \geq \pi_{tc}(wa') \vee \pi_{ts}(wa') \geq \pi_{tc}(wa)$.

$\mathbb{U}_S$ indicates the set of all possible schedules.

A resource allocation finds a schedule of scheduling instances using relevant scheduling parameters (cf. Definition 4). a set of parameters. The schedule produced by the resource allocation should cover all the scheduling instances.

Definition 7 (Resource Allocation): Let $P \subseteq \mathbb{U}_{pm}$ be the set of parameter names. A resource allocation defined over P, $ralc_P$, assigns schedules to the set of scheduling instances, i.e., $ralc_P \in \mathcal{P}(\mathbb{U}_{si}) \rightarrow \mathbb{U}_S$ such that, for any $SI \subseteq \mathbb{U}_{si}$, $\{\pi_{case}(wa) | wa \in ralc_P(SI))\} = \{\pi_{case}(si) | si \in SI\}$. $\mathbb{U}_{ralc,P}$ is the set of all possible such resource allocations.

We can quantify the quality of different schedules through the introduction of objective functions. Several common objective functions exist, including the total weighted completion time, total weighted tardiness, and maximum lateness [37]. The optimal schedule in terms of one objective function may not be the optimal one for the other objective functions.

Definition 8 (Objective Function): An objective function $obj \in \mathbb{U}_S \rightarrow \mathbb{R}^+$ assigns objective values to schedules. $\mathbb{U}_{obj}$ denotes the set of all possible objective functions.

An objective function is used to select the optimal resource allocation based on a specific objective function.

Definition 9 (Optimal Resource Allocation): Let $R_P \subseteq \mathbb{U}_{ralc,P}$ be the set of resource allocations of $P \subseteq \mathbb{U}_{pm}$ and $obj \in \mathbb{U}_{obj}$ the objective function. $ralc^*_{P,obj} \in R_P$ is optimal for obj if for any $ralc_P \in R_P$, $obj(ralc^*_{P,obj}(SI)) \leq obj(ralc_P(SI))$, where $SI \subseteq \mathbb{U}_{si}$.

IV. APPROACH

In this section, we introduce an approach to solve resource allocation problems of non-clairvoyant scheduling instances. Figure 3 presents the overview of the proposed approach consisting of two phases. The first phase is *offline prediction model construction* to build prediction models that determine the missing values of the parameters. If we aim to optimize the schedule of $SI \subseteq \mathbb{U}_{si}$ in terms of $obj \in \mathbb{U}_{obj}$ using a set of parameters $P \in \mathbb{U}_{pm}$, we first build prediction models to predict missing values of P in SI. In the first phase, we first extract features from historical event logs and train prediction models using the features. The second phase is *online resource scheduling* to optimize the schedule of SI using an optimal resource allocation based on a set of parameters P. In this phase, we first predict the missing parameter values of non-clairvoyant scheduling instances with the prediction models from the online phase and create enriched scheduling instances. Finally, we apply an optimization algorithm to find the optimal schedule of the enriched scheduling instances.

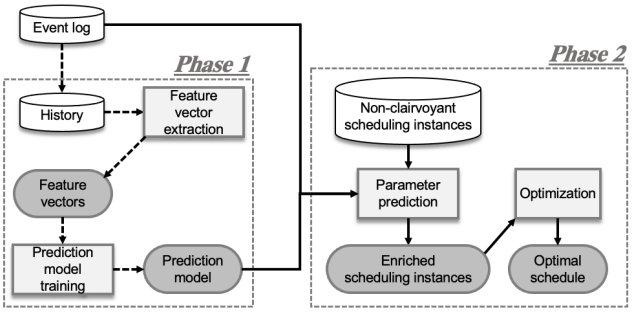


FIGURE 3. Overview of the proposed two-phase approach.

A. PHASE 1: OFFLINE PREDICTION MODEL CONSTRUCTION

A prediction model m_{pm} predicts the value of parameter $pm \in \mathbb{U}_{pm}$ for cases in business processes, while providing the uncertainty of the prediction to be used to evaluate if the prediction is reliable.

Definition 10 (Prediction Model): Let $UC \subseteq \mathbb{R}$ be the set of uncertainty values, i.e., $UC = \{x \in \mathbb{R} | 0 \leq x \leq 1\}$. A prediction model $model \in \mathbb{U}_{case} \rightarrow (\mathbb{U}_{val} \times UC)$ predicts the parameter value of cases, while providing the prediction uncertainty. $\mathbb{U}_{model}$ is the set of all possible prediction models.

We construct prediction models based on event logs. First, we need to generate feature vectors encoding characteristics of cases in business processes and the conditions of the process as independent variables and values of $pm \in \mathbb{U}_{pm}$ as independent variables. In this section, we abstract from the mechanism to create feature vectors from event logs. Instead, in section VI, we introduce a technique to extract *intra-case* features, representing the recent history of instances, and *inter-case* features, embedding the contextual information on the underlying business process.

Using feature vectors, we train prediction models. Various algorithms have been suggested and evaluated in the field of predictive business process monitoring [3]. In particular, algorithms based on deep neural networks have recently shown outperforming performances in various tasks (e.g., predicting next activity, predicting remaining time) [15]. In section VI, we instantiate the approach using a technique based on Bayesian Neural Networks (BNNs).

B. PHASE 2: ONLINE RESOURCE SCHEDULING

Using constructed prediction models, we predict the missing values of scheduling parameters of NoSIs. Since inaccurate predictions have a negative impact on achieving optimal objective values, we ignore the predictions with higher uncertainties. If the prediction is unreliable, we replace the missing value with the empty value so that the optimal resource allocation can ignore the prediction.

Definition 11 (Parameter Prediction): Let $P \subseteq \mathbb{U}_{pm}$ be a set of parameters. Let $model_{pm} \in \mathbb{U}_{model}$ be the prediction model that predicts values of parameter $pm \in P$ and $\tau_{pm} \in UC$ the uncertainty threshold for parameter $pm \in P$. $M(P) = \{model_{pm} \in \mathbb{U}_{model} | pm \in P\}$. $pred_{M(P)} \in \mathbb{U}_{si} \rightarrow \mathbb{U}_{si}$ predicts the missing parameter values for P in $(c, pmap) \in \mathbb{U}_{si}$, i.e., $pred_{M(P)}(si) = (c, pmap')$ such that, for any $pm \in P$ and $(val, uc) = model_{pm}(c)$,

$$pmap'(pm) = \begin{cases} pmap(pm) & \text{if } \pi_{pmap}(si) \neq \perp \\ val & \text{if } pmap(pm) = \perp \wedge uc \geq \tau_{pm} \\ \perp & \text{otherwise} \end{cases}$$

After completing missing values of P in the NoSIs by predicting them, we apply the optimal resource allocation defined over P to produce the pseudo-optimal schedule.

Definition 12 (Optimization): Let $\widehat{SI}$ denote $SI \subseteq \mathbb{U}_{si}$ whose missing parameters are enhanced by predictions, i.e., $\widehat{SI} = \{pred_{M(P')}(si) | si \in SI\}$. $optim_{P,obj}$ optimizes schedules of $SI \subseteq \mathbb{U}_{si}$ with respect to $obj \in \mathbb{U}_{obj}$, i.e., $optim_{P,obj} = ralc^{*}_{P,obj}(\widehat{SI})$.

V. PROBLEM: OPTIMIZING TWCT OF WEIGHT-AWARE NoSIs

This section introduces a resource allocation problem we endeavor to solve using the method developed by the proposed approach. The problem is to find the optimal resource allocation (cf. Definition 9) of *weight-aware NoSIs* (cf. Definition 4). The objective of the solution (cf. Definition 8) is *Total Weighted Completion Time (TWCT)*, and the constraints of the solution correspond to the constraints used to define schedules (cf. Definition 6).

Figure 4(a) presents the status of the *MRI* at time t in Figure 1 where scheduling instances with complete information in terms of the base set of parameters exist. The number annotated next to the patient indicates the symptom severity of the patient. The higher the number (i.e., weight) is, the more serious the symptom is. For example, p_4 who is planned to undergo *MRI* operation exhibits much more severe symptoms than those who are already in the waiting queue.

A. TOTAL WEIGHTED COMPLETION TIME (TWCT)

Optimizing schedules with respect to *TWCT* consists of two aspects. First, we should minimize the total completion time of scheduling instances. For example, we should assign p_5 to r_6 and p_6 to r_5 since r_6 and r_5 need less time to deal with p_5 and p_6, respectively, thus resulting in lower completion times of p_5 and p_6. Second, we should prioritize patients with a higher level of severity. Since p_4 has a higher severity level, we should reserve r_5 at time t to be assigned to p_4 at $t+1$ instead of assigning r_5 to p_6 at time t, even though it is more efficient in terms of the total completion time.

Figure 4(b) presents the resulting optimal schedule of Figure 4(a). Note that, to facilitate understanding, we focus on a single treatment (i.e., MRI) in a business process and only consider the next activity among other future activities as the influential factor when producing the schedule, whereas the optimal resource allocation defined over the base set of parameters provides a method for optimizing the schedules by comprehensively considering all the available information [37]. We refer readers to [37] for detailed explanations of the techniques for optimal resource allocation.

Definition 13 (Total Weighted Completion Time (TWCT)): Let $CT(c)$ be the completion time of $c \in \mathbb{U}_{case}$. Total weighted completion time $twct \in \mathbb{U}_{obj}$ is the objective function such that, for any $SI \subseteq \mathbb{U}_{si}$, $twct(SI) = \sum_{si \in SI} \pi_{pmap}(si)(weight) CT(\pi_{case}(si))$.

The schedule in Figure 4(b) has the TWCT of 27, adding 25 (i.e., $10 * 2 + 1 * 5$) and 2 (i.e., $1 * 2$).

B. WEIGHT-AWARE NoSI (W-NoSI)

In this problem, we assume that the scheduling instances have no information on the future activity sequence and the corresponding service times, as presented in Figure 4(c). They only have the complete information on the weight, with the information on the next activity available only after the completion of the current activity. For instance, we are unaware of how much time is required for r_5 and r_6 to operate on p_5. Moreover, we do not know which patients in the process will enter the *MRI* queue. We are only aware of the fact that p_5 and p_6 have the weight of 1, whereas p_4 has 10. After the completion of the current activity of p_4 (i.e., at time $t+1$), we recognize that this patient needs to undergo

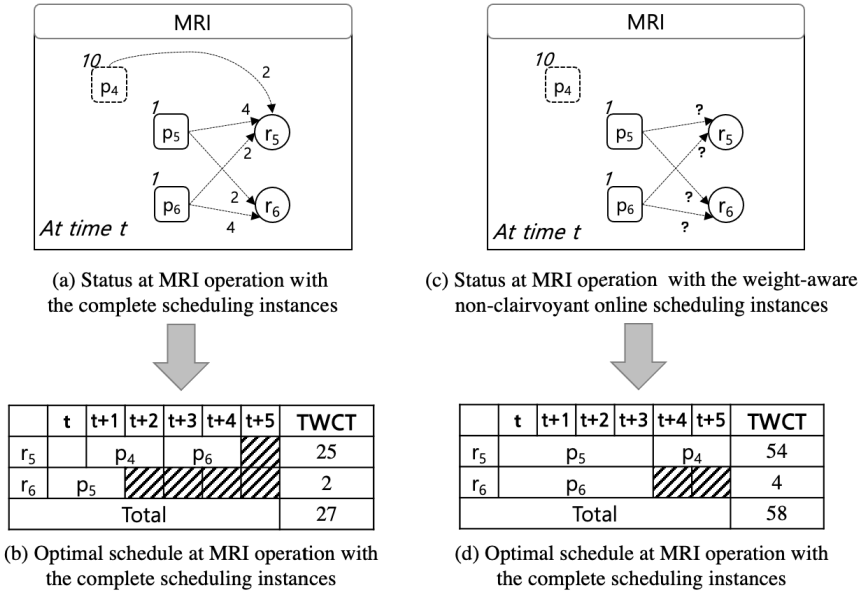


FIGURE 4. **Different status at *MRI operation* and corresponding optimal schedules.**

MRI. The scheduling instances are called *weight-aware NoSIs (W-NoSIs)*.

Suppose the activity sequence of p_5 in Figure 4(c) includes *triage*, *MRI*, and *evaluation* in order. The corresponding W-NoSI is represented by $wni_1 = (p_5, pmap)$, where $pmap(a_1) = triage$, $pmap(pt_{1,r1}) = 10$, $pmap(a_2) = MRI$, $pmap(pt_{2,r_5}) = \perp$, $pmap(pt_{2,r_6}) = \perp$, $pmap(a_3) = \perp$, and $pmap(weight) = 1$.

VI. IMPLEMENTING THE APPROACH: OPTIMIZING TWTC OF WEIGHT-AWARE NoSIs

Based on the approach proposed in section IV, we develop a method to solve the problem introduced in section V. More in detail, we aim to use the resource allocation that takes scheduling parameters, including 1) the next activity and 2) the service times of W-NoSIs. To this end, we predict the next activity and the service times of W-NoSIs and enhance W-NoSIs with the prediction. Next, we apply the optimal resource allocation defined over the parameters to minimize the TWCT. In the following, we introduce the rationale behind design choices to implement the approach. Next, we elaborate on each phase.

A. OFFLINE PREDICTION MODEL CONSTRUCTION

To implement the proposed method, we need two prediction models: one for predicting the next activity and another for predicting the service time of an activity by a resource. In this phase, we aim to construct the prediction models.

1) EXTRACTING FEATURE VECTORS

First, feature vectors are generated by feature encoding (*enc*) suggested in [38] that produces feature vectors considering contextual information in both knowledge-driven and data-driven manners. The feature encoding consists of two components: intra-case feature encoding (enc_{intra}), which encodes intra-case dependencies and inter-case feature encoding (enc_{inter}), which encodes inter-case dependencies.

The intra-case feature encoding $enc_{intra} \in \mathbb{U}^*_{event} \times \mathbb{N} \rightarrow \mathbb{U}^*_{event}$ generates the intra-case feature: $enc_{intra}(\sigma, k) = \langle e^{n-k+1}, \ldots, e^n \rangle$, where $\sigma = \langle e^1, \ldots, e^n \rangle$ and $k \leq n$. Suppose $\sigma = (e^1, e^2, e^3)$. $enc_{intra}(\sigma, 2) = (e^2, e^3)$ becomes the intra-case feature. The inter-case feature encoding $enc_{inter} \in \mathbb{U}^*_{event} \times \mathbb{U}_{EL} \rightarrow \mathbb{U}_{val}$ computes a feature given a trace and an event log. For instance, $enc^1_{inter}(\sigma, EL) = \gamma_1(\{\delta(\sigma) | \sigma \in EL\})$ returns the number of ongoing instances concurrently processed with σ based on $EL \in \mathbb{U}_{EL}$. For a detailed explanation and possible inter-case features, we refer readers to [38].

2) TRAINING PREDICTION MODELS

To implement the approach, we train the prediction models for the service time and the next activity prediction based on Bayesian Neural Networks (BNNs). In predictive business process monitoring, recent techniques use deep neural networks to achieve high accuracy [15]. However, the existing studies lack the explainability, especially the uncertainty of predictions, essential for assessing the reliability of the forecasts. BNNs enhance deep neural networks through the introduction of uncertainty to the networks.

Unlike traditional deep neural networks, wherein we aim to find point estimates for network parameters in the network, BNNs aim to find the posterior distribution of the parameters given a prior to them. However, it is particularly challenging to compute the posterior inference due to the nonlinearity and non-conjugacy of deep neural networks [39]. Moreover, traditional algorithms do not scale to a large number of parameters in deep neural networks.

In this regard, several studies have been suggested to approximate the posterior inference for BNNs, including variational Bayes [40] and probabilistic backpropagation [41]. However, each method requires different training methods that accordingly require non-trivial changes in the structure of the neural networks, thus impeding the use of recent techniques for predictive business process monitoring. Moreover, these approaches come with a prohibitive computational cost. For instance, to represent uncertainty, the number of parameters is doubled for the same network size.

In this method, we use the Monte Carlo dropout framework [42], [43], which requires trivial modifications in the existing architectures of deep neural networks, while providing uncertainty estimations in less computation time compared with other approximation methods. The basic idea of the Monte Carlo dropout framework is to apply stochastic *dropouts* [44], which drops out the units in the layer with probability after each hidden layer. The resulting model outputs are considered as random samples produced by the posterior distributions of the network parameters. The model uncertainty is calculated by the sample variance of the model outputs generated via multiple repetitions.

a: SERVICE TIME PREDICTION MODEL

$model_{st} \in \mathbb{U}_{model}$ is a function that returns the most likely service time of the current activity and the uncertainty of the prediction. Figure 5 presents an architecture of the service time prediction model. First, intra-case feature ft_{intra} is passed into LSTM layers to extract the sequential pattern that is internal to the case by considering the long-term dependency. Next, the information is concatenated with inter-case feature ft_{inter} that contains the contextual information outside the case. Note that we do not use the inter-case feature as an input for LSTM layers since we assume that the inter-case feature does not have sequential patterns that can be learned from LSTM. Finally, a fully connected layer produces the predicted service time $\hat{y}_i$ given the concatenated vector. In order to incorporate the Monte Carlo dropout framework, dropouts with probability p are applied to the layers of the neural network.

After repeating this procedure k times, we have k outputs, $\{\hat{y}_i^{(1)}, \ldots, \hat{y}_i^{(k)}\}$. The prediction is computed as sample mean of the outputs:

$$y_{st}^* = \frac{1}{k}\Sigma_{h=1}^{k}\hat{y}_i^{(h)} \tag{1}$$

The prediction uncertainty is approximated as sample variance of the outputs:

$$uc_{st} = \frac{1}{k}\Sigma_{h=1}^{n}(\hat{y}_i^{(h)} - y_{st}^*)^2 \tag{2}$$

The network weights are iteratively updated using the *Adam algorithm* [45] such that the mean absolute error between the actual service time and the predicted service time is minimized. As a regularization strategy to avoid overfitting of prediction models, we employ *Batch Normalization* [46].

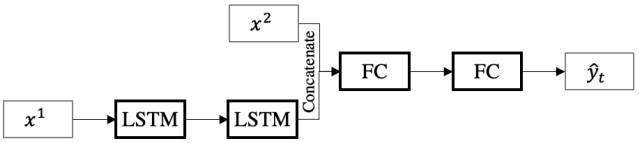


FIGURE 5. An architecture of the service time prediction model.

b: NEXT ACTIVITY PREDICTION MODEL

$model_{act} \in \mathbb{U}_{model}$ is a function that returns the most likely next activity and the uncertainty of the prediction. The architecture of the next activity prediction model follows the one of the service time prediction model. However, contrary to the processing prediction where the target variable is numerical, the target variable in the next activity prediction is categorical. The target variable is represented using one-hot encoding. Suppose A is the set of all possible next activities and $|A|$ indicates the size of the set and $A(i)$ the activity at index i. Given that $A = \{a, b, c\}$, $A(1) = a$, $A(2) = b$, and $A(3) = c$. It should be noted that we assume consistent ordering over the set of activities, e.g., $A(1)$ is always a. One-hot encoding of a, b, and c are represented as $[1, 0, 0]$, $[0, 1, 0]$, and $[0, 0, 1]$, respectively.

Similar to the service time prediction, we generate k model outputs, $\{\hat{y}_i^{(1)}, \ldots, \hat{y}_i^{(k)}\}$ where, for any $1 \leq m \leq k$, $\hat{y}_i^{(m)} = [\hat{y}_{i,1}^{(m)}, \ldots, \hat{y}_{i,|A|}^{(m)}]$ is a one-hot encoded vector such that each element represents the probability of the corresponding activity, e.g., $\hat{y}_{i,1}^{(m)}$ indicates the probability of $A(m)$ as the next activity.

The prediction is computed as the activity at the index that exhibits highest sample mean of the outputs:

$$y_{act}^* = A(m) \tag{3}$$

such that $m = arg\ max_{x \in \mathbb{N} \wedge 1 \leq x \leq |A|}(\bar{y}_{i,x})$, where $\bar{y}_{i,x} = \frac{1}{k}\Sigma_{h=1}^{k}\hat{y}_{i,x}^{(h)}$. The prediction uncertainty is approximated as sample variance of the outputs:

$$uc_{act} = \frac{1}{k}\Sigma_{h=1}^{k}(y_{i,m}^{(h)} - \bar{y}_{i,m})^2 \tag{4}$$

such that $\bar{y}_{i,m} = \frac{1}{k}\Sigma_{h=1}^{k}\hat{y}_{i,m}^{(h)}$. Moreover, we update the weights in such a way that we minimize the cross-entropy between the predicted next activity and the actual next activity.

B. ONLINE RESOURCE SCHEDULING

1) PARAMETER PREDICTION

Using the prediction model we constructed in the previous phase, we make predictions on the W-NoSIs. Once a scheduling instance completes the latest activity and waits for the next activity (i.e., a released instance), we predict the service times of its next activity by different resources using $model_{st}$. If a scheduling instance is currently under operations (i.e., a non-released instance), we first predict its most probable next activity using $model_{act}$ and then predict its service times in the same manner as for the released instance.

Algorithm 1 describes the prediction algorithm. If the scheduling instance si is released, as shown in lines 3-13, we predict the service times of si required by the responsible resources. The resulting service times by different resources are stored in ST. In the case of a non-released instance, as depicted in lines 14-28, we first predict the next activity and then the service times by different qualified resources. The predicted values are stored in ST. Note that only the predictions above thresholds (i.e., τ_{act} and τ_{st}) are considered.

Algorithm 1 Prediction

Require: W-NoSI si, event log EL, resource set R, service time prediction model $model_{st}$, next activity prediction model $model_{act}$, threshold for next activity prediction τ_{act}, threshold for service time prediction τ_{st}

Ensure: Predicted service times of si's next activity by different resources ST

1: $ST \leftarrow$ *empty dictionary*
2: $\sigma \leftarrow getTrace(si)$
3: **if** $isReleased(si)$ **then**
4: **for** $r \in R$ **do**
5: **if** $canHandle(r, getNextAct(si))$ **then**
6: $val_{st},\ ut_{st} \leftarrow model_{st}(si, EL)$
7: **if** $1 - ut_{st} \geq \tau_{st}$ **then**
8: $ST[r] \leftarrow val_{st}$
9: **else**
10: $ST[r] \leftarrow \perp$
11: **else**
12: $act_{next},\ ut_{act} \leftarrow model_{act}(si, EL)$
13: **if** $1 - ut_{act} \geq \tau_{act}$ **then**
14: **for** $r \in R$ **do**
15: **if** $canHandle(r, getNextAct(si))$ **then**
16: $p_j,\ ut_{st} \leftarrow model_{st}(si, EL)$
17: **if** $1 - ut_{st} \geq \tau_{st}$ **then**
18: $ST[r] \leftarrow val_{st}$
19: **else**
20: $ST[r] \leftarrow \perp$
21: **return** ST

For p_5 and p_6 in Figure 4(c), we predict the service times by r_5 and r_6. For p_4, we first predict the next activity (i.e., *MRI* operation) and then identify the resources who can perform the task (i.e., r_5). Subsequently, we predict the service time of the *MRI* by r_5. This manner enables us to enhance the NoSIs in Figure 4(c) to the complete scheduling instances that have information on the next activity and its service times by different resources, as presented in Figure 4(a).

The time complexity of Algorithm 1 depends on the complexity of BNNs used in the algorithm, and the complexity of a BNN depends on the specific MCMC algorithm used for inference. Assuming that the MCMC algorithm requires k samples and the time complexity of a single forward pass is $O(Ln^2)$, where L is the number of layers, and n is the number of neurons per layer, the total time complexity for a single input example is $O(MLn^2)$. Since we have R number of inputs, i.e., the number of resources, in the worst case, the algorithm's complexity is $O(RMLn^2)$.

2) OPTIMIZATION

For the optimization, we use the optimal resource allocation based on the *network simplex* method, which is computationally more efficient than the other possible approaches (e.g., integer linear programming and heuristic approaches including genetic algorithm), by guaranteeing to provide optimal solutions in polynomial time. To this end, after enhancing scheduling instances with predictions, we transform the resource allocation problem into the minimum cost and maximum flow network problem that aims at finding a maximum flow in the network with the smallest possible cost.

A network is a directed graph $G = (V, E)$, such that $E = V \times V$ with a source node $s \in V$ and a sink node $t \in V$, where each arc $(u, v) \in E$ has capacity $capa(u, v) > 0$, flow $fw(u, v) \geq 0$, and cost $cost(u, v) \in \mathbb{R}$. For any $(u, v) \in E$, $fw(u, v) \leq capa(u, v)$. The minimum cost and maximum flow network problem aim to maximize the total flow, i.e., $\Sigma_{(u,v)\in E}\ fw(u, v)$ while minimizing the total cost, i.e., $\Sigma_{(u,v)\in E}\ fw(u, v)capa(u, v)$.

First, we construct a bipartite graph, such that the nodes on the left denote the scheduling instances, $\widehat{SI}$, and those on the right denote the resources, R. Subsequently, we add an edge connecting a pair of nodes in the bipartite graph, i.e., $(si \in \widehat{SI}, r \in R)$ if the scheduling instance si can be processed by the resource r. Note that, using the next activity predictions in the previous step, we can add edges connecting the non-released instances and the qualified resources.

We annotate each edge with $(cost, capa)$, such that $cost = \frac{st+rem(si),rem(r),0)}{w}$ where st denotes the service time of si's activity by resource r, $rem(si)$ denotes the remaining time of si's current activity, and $rem(r)$ denotes the remaining time for r to be ready, and w denotes the weight of si. Note that the cost function is designed to optimize our objective (i.e., minimizing the TWCT), i.e., lower cost is assigned to edges with less service times and higher instance weights, whereas higher cost is given if the instance or resource is not prepared. Finally, we solve the minimum cost maximum flow problem to find the optimal schedule, using the network simplex method [47].

Algorithm 2 generates the optimal schedule of the given instances and resources. Both released instances and predicted non-released instances (i.e.,non-released instances in which the uncertainties of predictions are above the given thresholds), $\widehat{SI}$, are instantiated to the set of nodes on the left-hand side of a bipartite graph. The resources, R, are instantiated to the set of nodes on the right-hand side of the bipartite graph. As shown in lines 1-7, we first produce a source node and a sink node and then add edges connecting the source node to the left and right nodes to the sink node with $(cost = 0, capa = 1)$. Next, we add edges connecting the left nodes and right nodes if the resources can process the instances with $(cost, capa = 1)$ such that $cost = \frac{pt_j+max(rem(si),rem(r),0)}{w}$.

Algorithm 2 Resource Scheduling Algorithm

Require: set of scheduling instances $\widehat{SI}$, set of resources R, event log EL, set of service times ST
Ensure: Optimal Schedule M

Produce source node s, sink node t;
for *node si* $\in \widehat{SI}$ **do**
  add edge $(s, si, (0, 1))$
for *node r* $\in R$ **do**
  add edge $(r, t, (0, 1))$
for *node si* $\in \widehat{SI}$ **do**
  $ST \leftarrow predict(si, EL)$
  for *node r* $\in R$ **do**
    if $\exists\, r \in getKeys(ST)$ **then**
      $cost \leftarrow \frac{st+max(rem(si),rem(r),0)}{w}$
      add edge $(si, r, (cost, 1))$
$M \leftarrow NetworkSimplex(S, T)$
return M

Figure 6 shows the minimum cost and maximum flow network problem that represents the situation depicted in Figure 4(c). In the network, $\widehat{SI}$ has three elements, i.e., $\widehat{SI} = \{p_4, p_5, p_6\}$ where p_4 is a non-released instance and p_5, p_6 are released instances. R has two elements, i.e., r_5, r_6, all ready for the assignments. The arcs are annotated with the corresponding $(cost, capa)$. The network simplex method returns an optimal schedule at time t, where the optimal matches are indicated by bold lines, i.e., p_4 to r_5, p_5 to r_6.

The time complexity of Algorithm 2 relies on the complexity of the network simplex algorithm. The worst-case time complexity of the algorithm is $O(VElogV)$, where V is the number of nodes and E is the number of edges [48]. However, in practice, the algorithm performs better than its worst-case complexity suggests, e.g., by exploiting the sparsity of the network.

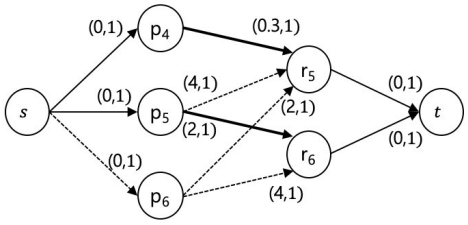


FIGURE 6. An example of resource scheduling.

We identify executable and non-executable matches from the optimal schedule. An executable match indicates that both the corresponding instance and resource are available at the current time, whereas a non-executable match indicates that either of them is not available. In Figure 6, p_5 and r_6 have an executable match, whereas p_4 and p_5 have a non-executable match. We only execute executable matches, i.e., p_5 and r_6.

VII. EVALUATION

This section evaluates the proposed method using a synthetic event log and a real-life event log. The evaluation objectives are twofold: 1) evaluating whether the proposed method produces the schedules of W-NoSIs that optimize TWCT at a reasonable computation time and 2) evaluating the effects of the inaccuracy of the prediction on the scheduling performance. Based on our objectives, we devise the following research questions (RQs):

- RQ1: Does the proposed method improve the performance of TWCT optimization compared to a baseline approach?
- RQ2: Does the proposed method impede the computation time compared to a baseline approach?
- RQ3: Does the accuracy of predictions in the proposed method affect the performance of TWCT optimization?

To answer the first two questions, we compare our proposed method with the optimal resource allocation defined over the already-known information (i.e., the weight and the next activity if the instance is released) without any predictions, both in terms of the scheduling performance and computation time. In section VII-A, we elaborate this approach as a baseline. For RQ3, we conduct experiments by using the proposed method, but replacing the predictions with random values generated by adding some noise to the actual values. By changing the amount of noise, we measure the impact of inaccurate predictions.

For the prediction model construction, we initialized all the network weights using a uniform random distribution over $[-0.1, 0.1]$. The models were trained with a batch size of 16 with 100 epochs and a learning rate of 0.001. When a metric stops improving, the learning rate decreases by a factor of 0.1. The training is stopped if the loss stops decreasing for five epochs.

The implementations are written in Python (version 3.6.2), and the source code is publicly available at *GitHub repository*.[1] The experiments were conducted using a quad-core 7th-Generation Intel Core i5 processor with 32GB of RAM.

A. A BASELINE APPROACH

Given that we do not adopt the proposed approach, the best alternative approach is the maximum exploitation of the provided information. By definition of W-NoSI, we have the information on the weight of a scheduling instance and its next activity if it is already released. Considering the given information, assigning the scheduling instances with higher weights earlier than the ones with lower weights is the optimal way to produce a schedule. This means that we should find the schedule that maximizes the total weights of the assigned instances.

For instance, there is no difference between assigning 1) p_5 to r_5 and p_6 to r_6 and 2) p_5 to r_5 and p_6 to r_6, i.e., in any case, assignments have the total weights of 2. With the given information, it is expected that 1) and 2) have the same cost. However, assume that there exists one more p' in Figure 4(c) that has the weight of 8 and requires resource r_6. In this case, we should assign p' to r_6 and p_5 or p_6 to r_5

[1] https://github.com/gyunamister/prediction_based_resource_allocation

(i.e., the total weights of 11), since the waiting time of p' (i.e., higher completion time of p' as a consequence) contributes to the higher objective.

Since our problem now is to find the maximum total weights, we can represent it as the minimum-cost and maximum-flow problem as in Section V, and find the optimal solution by solving the problem. However, contrary to the proposed method, we use the additive inverse of weight w (i.e., $-w$) as the cost function, since maximizing the weights corresponds to minimizing their additive inverses.

B. EXPERIMENT ON AN ARTIFICIAL HEALTHCARE PROCESS

1) EXPERIMENTAL DESIGN

For the experiment, we designed a simplified process in the emergency department of a hospital. The process comprises 11 activities and 25 resources. The resources have their own set of activities to perform, and the service times for the activities depend on their proficiency level. Patients with different priorities ranging from 1 to 10 arrive regularly at the process. Assuming a non-clairvoyant environment, we are unaware of when a patient is ready for the next operation. Also, the patient's next activity is known after the completion of the patient's current activity.

For the construction of prediction models, we simulate the process for seven days and generate the event log containing 6,575 events and 1,000 instances. For the resource allocation, we use a set of scheduling instances (i.e., patients) that enter the process at a regular interval. To evaluate the proposed method in different business environments, we create five sets, involving 40, 60, 80, 100, and 120 patients, respectively. The patients in each set are assumed to arrive at equal intervals for 3 h. For instance, a set of 40 patients enter the process at every 3/40 h.

We answer RQ1 and RQ2 by comparing the TWCTs and the computation times of the proposed method and the baseline approach with different numbers of patients for resource allocation. To answer RQ3, we analyze the TWCT by varying the accuracy of the predictions from 0% to 100%.

2) RESULTS

Figure 7(a) presents the results related to RQ1 with the sets of 80, 90, 100, 110, 120, 130, 140, 150 and 160 patients. In all cases of different sets, the proposed method outperforms the baseline approach at optimizing schedules in terms of the TWCT since it identifies the most efficient resources and considers potential instances when producing schedules. For example, when the number of patients is 100, the TWCT of the baseline approach (i.e., 6,348) is 14% higher than that of the proposed method (i.e., 5,420).

Regarding RQ2, Figure 7(b) presents the computation time required for the proposed method and the baseline approach. The computation time of the proposed method is relatively higher than that of the baseline approach. The higher computation time results from computing predictions. However, as the underlying process lasts 3 h, the total computation time required (i.e., less than 80 s) is suitable for the real-time use of the proposed method.

Figure 8 presents the effect of prediction accuracy on scheduling accuracy. The experiment emulates X% accuracy in the service time prediction through the provision of an absolute percentage error of $(100-X)$% in the actual service time. For example, given the accuracy of 80%, it is predicted that the actual service time of 10 in the experiment is $8(=10-2)$ or $12(=10+2)$. We also imitate the accuracy of X% in the next activity prediction using the X% of predictions with the actual next activity and $(100-X)$% with the incorrect one. For instance, the accuracy of 80% means that 80% of the next activity predictions are the actual next activities, and 20% of the predictions are provided with the incorrect activities.

As presented in Figure 8, a huge difference is observed in the TWCT between poor prediction accuracy (i.e., 0% to 40%) and fairly good prediction accuracy (i.e., 60% to 100%). In the case of 110 patients, the TWCT with the prediction accuracy of 0% to 40% is about 8,000, whereas the one with the prediction accuracy of 60% to 100% is about 6,000. As the prediction accuracy increases from 40% to 60%, the scheduling performance also improves significantly. For example, the scheduling performance improves from 7670 (with 40% accuracy) to 6036 (with 60% accuracy).

The dotted line indicates the TWCT produced by the baseline approach. In most cases, the objective value is lower than that of the baseline approach when achieving 60% of the prediction accuracy. In the worst case, i.e., when the number of patients is 130, 80% of the prediction accuracy is required to reach the objective value lower than the baseline approach, whereas the best case is when the number of patients is 100, where the lower objective is achieved with just 50% of the accuracy.

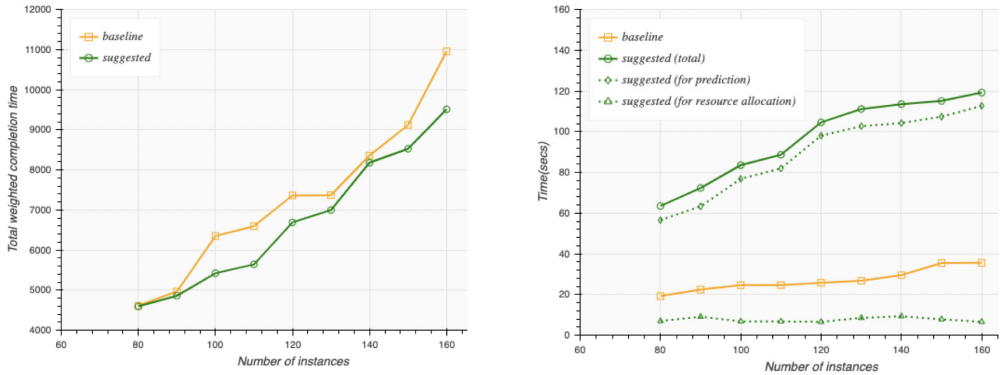
Note that the prediction accuracy and TWCT do not have positive relationships in all cases. It is due to limited information, i.e., we still do not know the whole future sequence of patients; thus, the efforts to greedily find local optimal solutions do not lead to the global optimum solution.

C. EXPERIMENT ON A REAL-LIFE HEALTHCARE PROCESS

1) EXPERIMENTAL DESIGN

This experiment uses the real-life event log from the emergency department in a tertiary hospital in South Korea. The underlying process of the event log consists of 18 activities, which are operated by 754 resources who work in a shift system. The event log contains 421,995 events by 27,478 patients who entered the emergency department to perform the activities from January 1, 2018, to November 30, 2018. Each patient has a numerical indicator describing the symptom severity.

We create the historical event log used for the prediction model construction by filtering the events before October 2018. Then, we generate the set of scheduling instances for



(a) Total weighted completion time of varying number of scheduling instances

(b) Computation time of varying number of scheduling instances

FIGURE 7. Total weighted completion time and computation time of varying $|C|$.

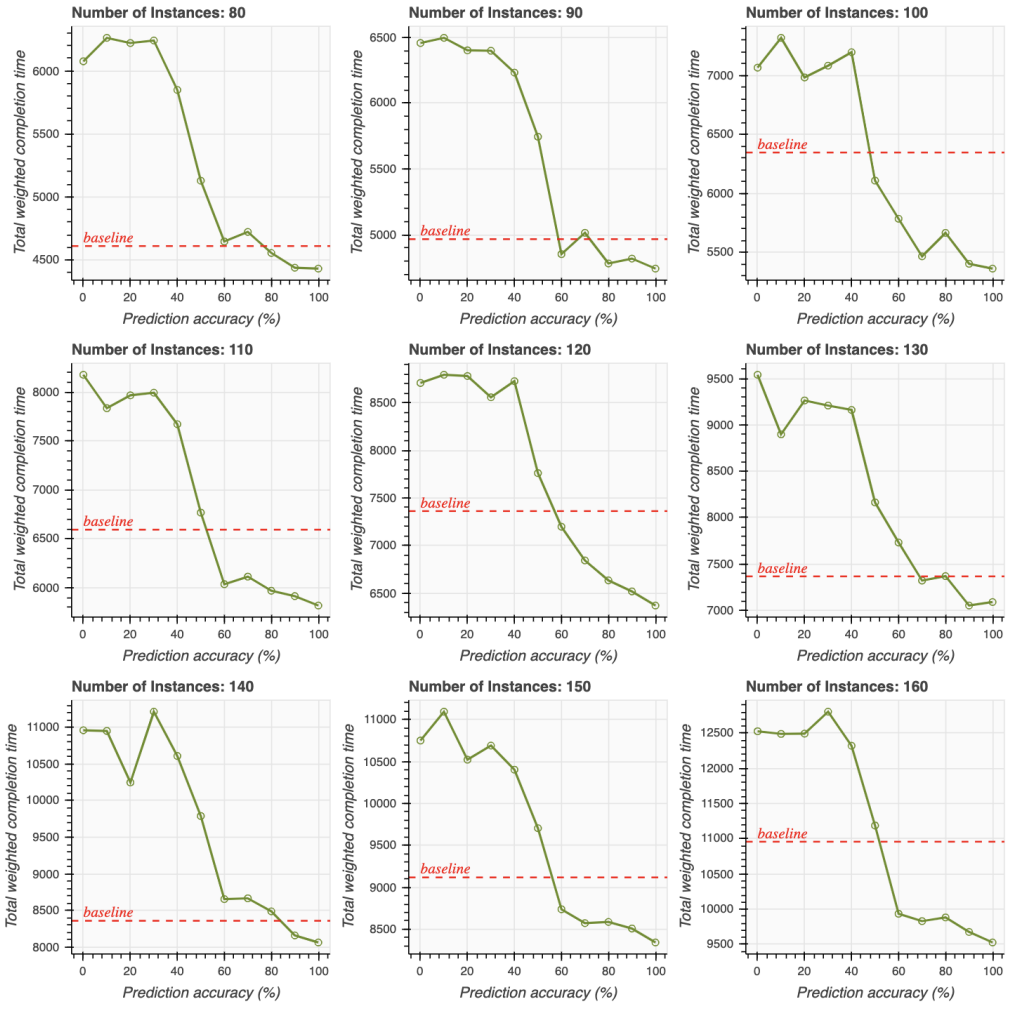
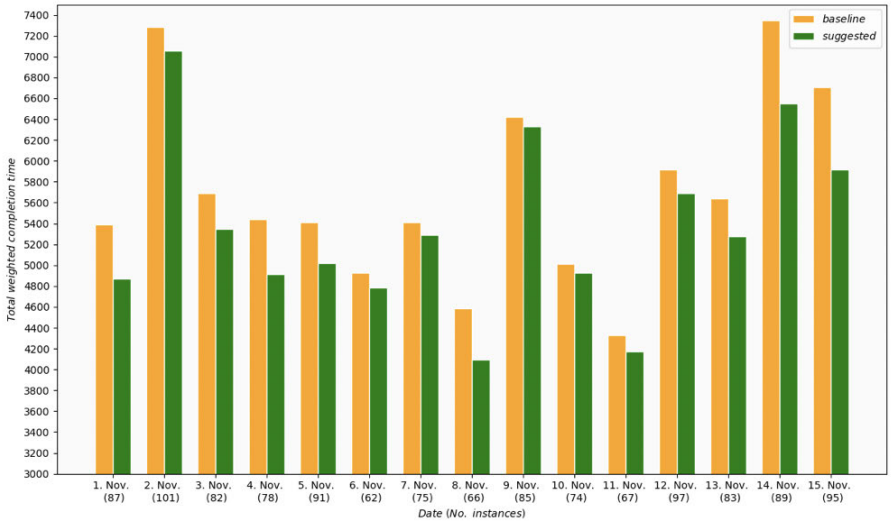
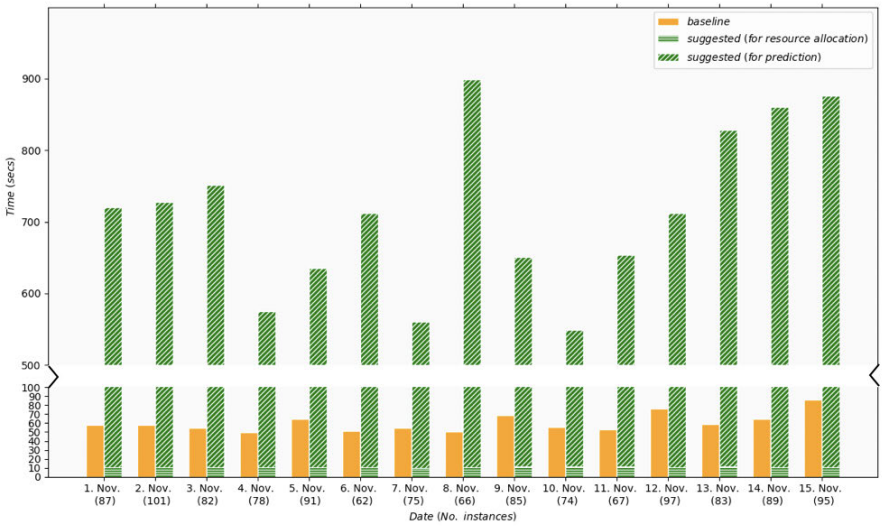


FIGURE 8. Effect of the prediction accuracy on the total weighted completion time.

each date from November 1 to 15, 2018 using the information on the patients who entered the process on each date. In other words, we have 15 sets of scheduling instances, where each set consists of the patients who appeared on each date. For example, the set for November 1, 2018, is composed of 87 scheduling instances requiring resource allocation. We scale the indicator for the symptom severity to the range between 0 and 10. As in the previous experiment, we do not



(a) Total weighted completion time of the schedule for each date



(b) Computation time of the schedule for each date

FIGURE 9. Total weighted computation time and computation time for each date.

have the information on the exact service times of activities by different resources unless recorded in the event log. Thus, we use the estimation models to estimate the expected service times by different possible resources. Such models are built from the historical log separately from the ones used in the proposed method.

2) RESULTS

Figure 9 presents the experimental results that compare the proposed method and the baseline approach both in terms of the scheduling performance and the computation time. The proposed method outperforms the baseline approach, for any set of scheduling instances. For example, on November 1, 2018, the proposed method has the TWCT of 4871, whereas the baseline approach has 5386. This means that the proposed method improves the objective by 9.6% compared with the baseline approach.

However, the proposed method has a much higher computation time compared with the baseline approach. This is due to the high number of predictions generated by the proposed method to find the most efficient resources among the possible resources. For example, out of the 721.6 seconds spent on the resource allocation on March 01, 2018, 710.4 seconds were spent on the predictions. However, considering that we produce schedules for the whole day, a few minutes of computation does not hinder the application of the proposed method.

VIII. DISCUSSION

In the evaluation section, we have presented the effectiveness and efficiency of the method on one artificial log and two

real-life logs. The proposed method significantly reduces the TWCT by allocating the most efficient resource to the instance and saving some resources for cases with higher priorities when they are predicted to be released. Although making predictions requires additional computations resulting in a relatively high computation time, the experiments have demonstrated that it is still feasible to use the method in practice. Moreover, the experiments indicate that the proposed method produces stable scheduling performances at a certain level of prediction accuracy.

Our work has important implications for both research and practice. From a research perspective, it combines the predictive business process monitoring in process mining and the optimization in operations research to improve business processes by optimizing resource allocation in a non-clairvoyant business environment. From a practical perspective, our proposed method provides practitioners with an efficient method for optimizing resource allocation in their business processes based on novel predictions.

This work has several limitations. The proposed approach depends on the accuracy of a prediction model, making the construction of accurate prediction models preliminary to applying the approach. Experiments have demonstrated that inaccurate prediction of the service time of an instance hinders finding the instance with the most efficient resource. Moreover, the evaluation in real-life logs has the following limitation: we do not have complete information (i.e., service times by different alternative resources) to simulate the experiments. Alternatively, we estimated the required information using estimation models that produce it as close as possible to reality; however, there is no guarantee that the estimated values are the same as the reality.

IX. CONCLUSION

This paper suggested a novel two-phase approach to improve a business process using prediction results from predictive business process monitoring techniques. We specifically focused on non-clairvoyant resource allocation problems in business processes where we have limited information on the scheduling instances, for which we strive to optimize the schedules. The first phase aims to construct prediction models predicting the values of parameters defined in the scheduling instances. The second phase optimizes the schedule of the scheduling instances by predicting the missing values of the parameters and applying a technique for optimal resource allocation. To evaluate the effectiveness of the proposed approach, we defined a specific non-clairvoyant resource allocation problem, where we can optimize the schedules of weight-aware non-clairvoyant scheduling instances in terms of the TWCT. Based on the proposed approach, we developed a method for solving the problem by incorporating the BNNs into the minimum cost and maximum flow algorithm.

In future work, we plan to extend the two-phase approach so that other goals in business processes can be achieved, aside from optimizing resource allocation, such as minimizing the potential risks in the business process. Another direction of future work is to apply the proposed approach to develop methods that solve other non-clairvoyant resource allocation problems with different scheduling instances and objectives. Also, we plan to deal with *concept drift* in the approach. As presented in the evaluation, the efficiency of the proposed approach depends on the construction of accurate prediction models. Since the business environment has a dynamic nature, the prediction models should adapt to the environment's gradual and incremental changes (i.e., concept drift).

REFERENCES

[1] L. Reinkemeyer, Ed., *Process Mining in Action: Principles, Use Cases and Outlook*. Cham, Switzerland: Springer, 2020.
[2] M. Polato, A. Sperduti, A. Burattin, and M. de Leoni, "Time and activity sequence prediction of business process instances," *Computing*, vol. 100, no. 9, pp. 1005–1031, 2018.
[3] A. E. Márquez-Chamorro, M. Resinas, and A. Ruiz-Cortés, "Predictive monitoring of business processes: A survey," *IEEE Trans. Services Comput.*, vol. 11, no. 6, pp. 962–977, Nov./Dec. 2018.
[4] W. M. P. van der Aalst, M. H. Schonenberg, and M. Song, "Time prediction based on process mining," *Inf. Syst.*, vol. 36, no. 2, pp. 450–475, Apr. 2011.
[5] W. M. P. van der Aalst, M. La Rosa, and F. M. Santoro, "Business process management: Don't forget to improve the process!" *Bus. Inf. Syst. Eng.*, vol. 58, no. 1, pp. 1–6, Feb. 2016.
[6] D. Bertsimas and N. Kallus, "From predictive to prescriptive analytics," 2014, *arXiv:1402.5481*.
[7] M. Arias, R. Saavedra, M. R. Marques, J. Munoz-Gama, and M. Sepúlveda, "Human resource allocation in business process management and process mining: A systematic mapping study," *Manage. Decis.*, vol. 56, no. 2, pp. 376–405, Feb. 2018.
[8] W. Zhao and X. Zhao, "Process mining from the organizational perspective," in *Foundations of Intelligent Systems*, Z. Wen and T. Li, Eds. Berlin, Germany: Springer, 2014, pp. 701–708.
[9] C. Tsiushuang, "Dispatching rules for manufacturing job-shop operations," *IFAC Proc. Volumes*, vol. 26, no. 2, pp. 975–978, Jul. 1993.
[10] N. Megow, "Coping with incomplete information in scheduling—Stochastic and online models," in *Operations Research Proceedings 2007*, J. Kalcsics and S. Nickel, Eds. Berlin, Germany: Springer, 2007, pp. 17–22.
[11] J. R. Wieringa, *Design Science Methodology for Information Systems and Software Engineering*. Cham, Switzerland: Springer, 2014.
[12] K. Peffers, T. Tuunanen, and B. Niehaves, "Design science research genres: Introduction to the special issue on exemplars and criteria for applicable design science research," *Eur. J. Inf. Syst.*, vol. 27, no. 2, pp. 129–139, Mar. 2018.
[13] F. Folino, M. Guarascio, and L. Pontieri, "Discovering context-aware models for predicting business process performances," in *On the Move to Meaningful Internet Systems: OTM 2012*, R. Meersman, H. Panetto, T. Dillon, S. Rinderle-Ma, P. Dadam, X. Zhou, S. Pearson, A. Ferscha, S. Bergamaschi, and I. F. Cruz, Eds. Berlin, Germany: Springer, 2012, pp. 287–304.
[14] J. Evermann, J.-R. Rehse, and P. Fettke, "Predicting process behaviour using deep learning," *Decis. Support Syst.*, vol. 100, pp. 129–140, Aug. 2017.
[15] N. Tax, I. Verenich, M. L. Rosa, and M. Dumas, "Predictive business process monitoring with LSTM neural networks," in *Advanced Information Systems Engineering*, E. Dubois and K. Pohl, Eds. Cham, Switzerland: Springer, 2017, pp. 477–492.
[16] N. Mehdiyev, J. Evermann, and P. Fettke, "A novel business process prediction model using a deep learning method," *Bus. Inf. Syst. Eng.*, vol. 62, no. 2, pp. 143–157, Apr. 2020.
[17] D. Castelvecchi, "Can we open the black box of AI," *Nature*, vol. 538, no. 7623, pp. 20–23, 2016.
[18] M. Stierle, S. Weinzierl, M. Harl, and M. Matzner, "A technique for determining relevance scores of process activities using graph-based neural networks," *Decis. Support Syst.*, vol. 144, May 2021, Art. no. 113511.
[19] I. Teinemaa, M. Dumas, F. M. Maggi, and C. D. Francescomarino, "Predictive business process monitoring with structured and unstructured data," in *Business Process Management*, M. L. Rosa, P. Loos, and O. Pastor, Eds. Cham, Switzerland: Springer, 2016, pp. 401–417.

[20] R. Conforti, M. de Leoni, M. La Rosa, W. M. P. van der Aalst, and A. H. M. ter Hofstede, "A recommendation system for predicting risks across multiple business process instances," *Decis. Support Syst.*, vol. 69, pp. 1–19, Jan. 2015.

[21] A. S. Fahrenkrog-Petersen, N. Tax, I. Teinemaa, M. Dumas, M. D. Leoni, F. M. Maggi, and M. Weidlich, "Fire now, fire later: Alarm-based systems for prescriptive process monitoring," 2019, *arXiv:1905.09568*.

[22] S. Weinzierl, S. Zilker, M. Stierle, M. Matzner, and G. Park, "From predictive to prescriptive process monitoring: Recommending the next best actions instead of calculating the next most likely events," in *Proc. Entwicklungen, Chancen und Herausforderungen der Digitalisierung, Internationalen Tagung Wirtschaftsinformatik, (WI)*, N. Gronau, M. Heine, H. Krasnova, K. Poustcchi, Eds. Potsdam, Germany, Mar. 2020, pp. 364–368. GITO Verlag, 2020.

[23] G. Park, M. Comuzzi, and M. P. Wil van der Aalst, "Analyzing process-aware information system updates using digital twins of organizations," in *Proc. Int. Conf. Res. Challenges Inf. Sci.* in Lecture Notes in Business Information Processing Book Series, vol. 446, R. S. S. Guizzardi, J. Ralyté, and X. Franch, Eds. Barcelona, Spain: Springer, May 2022, pp. 159–176.

[24] M. Dees, M. D. Leoni, M. P. Wil van der Aalst, and A. Hajo Reijers, "What if process predictions are not followed by good recommendations," in *Proc. Ind. Forum BPM Co-Located With 17th Int. Conf. Bus. Process Manag. (BPM)*, vol. 2428, J. vom Brocke, J. Mendling, M. Rosemann, Eds. Vienna, Austria, Sep. 2019, pp. 61–72.

[25] M. D. Leoni, M. Dees, and L. Reulink, "Design and evaluation of a process-aware recommender system based on prescriptive analytics," in *Proc. 2nd Int. Conf. Process Mining (ICPM)*, Oct. 2020, pp. 9–16.

[26] P. Badakhshan, G. Bernhart, J. Geyer-Klingeberg, J. Nakladal, S. Schenk, and T. Vogelgesang, "The action engine—Turning process insights into action," in *Proc. ICPM Demo Track*, Aachen, Germany, 2019, pp. 28–31.

[27] G. Park and W. M. P. van der Aalst, "Action-oriented process mining: Bridging the gap between insights and actions," *Prog. Artif. Intell.*, 2022. [Online]. Available: https://link.springer.com/10.1007/s13748-022-00281-7, doi: 10.1007/s13748-022-00281-7.

[28] G. Park and W. M. P. Van Der Aalst, "Realizing a digital twin of an organization using action-oriented process mining," in *Proc. 3rd Int. Conf. Process Mining (ICPM)*, Oct. 2021, pp. 104–111.

[29] G. Park and M. Song, "Prediction-based resource allocation using LSTM and minimum cost and maximum flow algorithm," in *Proc. Int. Conf. Process Mining (ICPM)*, Jun. 2019, pp. 121–128.

[30] M. A. Cruz-Chavez, M. H. C. Rosales, J. C. Zavala-Diaz, J. A. H. Aguilar, A. Rodriguez-Leon, J. C. P. Avelino, M. E. L. Ortiz, and O. H. Salinas, "Hybrid micro genetic multi-population algorithm with collective communication for the job shop scheduling problem," *IEEE Access*, vol. 7, pp. 82358–82376, 2019.

[31] C.-L. Liu, C.-C. Chang, and C.-J. Tseng, "Actor-critic deep reinforcement learning for solving job shop scheduling problems," *IEEE Access*, vol. 8, pp. 71752–71762, 2020.

[32] J. Błażewicz, W. Domschke, and E. Pesch, "The job shop scheduling problem: Conventional and new solution techniques," *Eur. J. Oper. Res.*, vol. 93, no. 1, pp. 1–33, Aug. 1996.

[33] J. H. Blackstone, D. T. Phillips, and G. L. Hogg, "A state-of-the-art survey of dispatching rules for manufacturing job shop operations," *Int. J. Prod. Res.*, vol. 20, no. 1, pp. 27–45, 1982.

[34] X. Qiu and H. Y. K. Lau, "An AIS-based hybrid algorithm with PDRs for multi-objective dynamic online job shop scheduling problem," *Appl. Soft Comput.*, vol. 13, no. 3, pp. 1340–1351, Mar. 2013.

[35] L. Becchetti and S. Leonardi, "Non-clairvoyant scheduling to minimize the average flow time on single and parallel machines," in *Proc. 33d Annu. ACM Symp. Theory Comput.*, Jul. 2001, pp. 94–103.

[36] S. Im, J. Kulkarni, and K. Munagala, "Competitive algorithms from competitive equilibria: Non-clairvoyant scheduling under polyhedral constraints," *J. ACM*, vol. 65, no. 1, pp. 1–33, 2018.

[37] L. M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 3rd ed. New York, NY, USA: Springer, 2008.

[38] A. Senderovich, C. D. Francescomarino, and F. M. Maggi, "From knowledge-driven to data-driven inter-case feature encoding in predictive process monitoring," *Inf. Syst.*, vol. 84, pp. 255–264, Sep. 2019.

[39] M. R. Neal, *Bayesian Learning for Neural Networks*. New York, NY, USA: Springer, 1996.

[40] A. Wu, S. Nowozin, E. Meeds, E. R. Turner, J. M. Hernández-Lobato, and L. A. Gaunt, "Fixing variational Bayes: Deterministic variational inference for Bayesian neural networks," 2018, *arXiv:1810.03958*.

[41] J. M. Hernández-Lobato and R. P. Adams, "Probabilistic backpropagation for scalable learning of Bayesian neural networks," in *Proc. 32nd Int. Conf. Mach. Learn.*, vol. 37, 2015, pp. 1861–1869.

[42] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *Proc. 33rd Int. Conf. Mach. Learn. (PMLR)*, vol. 48, 2016, pp. 1050–1059.

[43] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 1027–1035.

[44] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[45] P. D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent., (ICLR)*, San Diego, CA, USA, 2015, pp. 1–15.

[46] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, vol. 37, Jul. 2015, pp. 448–456.

[47] W. H. Cunningham, "A network simplex method," *Math. Program.*, vol. 11, no. 1, pp. 105–116, Dec. 1976.

[48] R. E. Tarjan, "Dynamic trees as search trees via Euler tours, applied to the network simplex algorithm," *Math. Program.*, vol. 78, no. 2, pp. 169–177, Aug. 1997.

GYUNAM PARK received the M.S. degree from the Department of Industrial and Management Engineering, Pohang University of Science and Technology (POSTECH), Pohang, South Korea, in 2019. He is currently pursuing the Ph.D. degree with the Department of Computer Science, RWTH Aachen University, Aachen, Germany. He has published scientific papers in journals, such as *Decision Support Systems* and *Progress in Artificial Intelligence* and conferences, including Business Process Management and International Conference on Process Mining. His research interests include process mining, data science, online operational support, machine learning, and deep learning.

MINSEOK SONG (Member, IEEE) received the Ph.D. degree from the Department of Industrial and Management Engineering, Pohang University of Science and Technology (POSTECH), in 2006. He stayed with the Information Systems Group, Department of Industrial Engineering and Innovation Sciences, Eindhoven University of Technology, as a Postdoctoral Researcher, from 2006 to 2009. He was an Assistant Professor/an Associate Professor with the Ulsan National Institute of Science and Technology (UNIST), from 2010 to 2015. He is currently an Associate Professor with the Department of Industrial and Management Engineering, POSTECH. He has published more than 100 scientific articles in several top-level venues, such as *Decision Support Systems*, *Information Systems*, *Journal of Information Technology*, and *International Journal of Medical Informatics*. His research interests include business process management, process mining, business analytics, simulation, and health information systems.