Prediction-based Resource Allocation using LSTM and minimum cost and maximum flow algorithm

Gyunam Park and Minseok Song
Department of Industrial and Management Engineering
POSTECH (Pohang University of Science and Technology)
77 Cheongam-Ro. Nam-Gu. Pohang. South Korea 37673
{gnpark, mssong}@postech.ac.kr

***Abstract*—Predictive business process monitoring aims at providing the predictions about running instances by analyzing logs of completed cases of a business process. Recently, a lot of research focuses on increasing productivity and efficiency in a business process by forecasting potential problems during its executions. However, most of the studies lack suggesting concrete actions to improve the process. They leave it up to the subjective judgment of a user. In this paper, we propose a novel method to connect the results from predictive business process monitoring to actual business process improvements. More in detail, we optimize the resource allocation in a non-clairvoyant online environment, where we have limited information required for scheduling, by exploiting the predictions. The proposed method integrates offline prediction model construction that predicts the processing time and the next activity of an ongoing instance using LSTM with online resource allocation that is extended from the minimum cost and maximum flow algorithm. To validate the proposed method, we performed experiments using an artificial event log and a real-life event log from a global financial organization.**

I. Introduction

Process mining techniques allow the extraction of in-depth insights in process-related problems, which business corporations face, from event logs available in Process-Aware Information Systems(PAISs) [1]. Through the application of process mining, companies can discover the process model describing how they work, examine the difference between a reference process model and a discovered one, and calculate a variety of performance measures such as case duration, resource utilization, and bottlenecks in processes [2]. These techniques are commonly applied in an offline fashion where only completed cases are being considered. Recently, however, online process mining techniques, which considers running cases, are gaining more interests [3].

In process mining, predictive business process monitoring provides the predictions concerning the future status of ongoing cases of a business process [4]. It aims at improving business processes by offering timely information that enables proactive and corrective actions [5]. Previous studies focus on the prediction tasks (e.g., time, risk probability, performance indicators, and next event) using several methods such as annotated transition system, machine learning, or statistics [5], [6], [7].

The previous studies, however, do not suggest how the prediction results can be exploited to improve business processes, leaving it up to the subjective judgment of a user [2]. In order to achieve the goal of process improvement, the prediction results should be transformed into concrete improvement actions. In this regard, we demonstrate how the results from predictive process monitoring can be transformed to suggest an actionable recommendation on resource allocation aiming at improving business processes.

Resource allocation is an essential and challenging problem in business process management (BPM) [8]. Efficient resource allocation improves productivity, balances resource usage, and reduces execution costs [9]. Resource allocation in BPM shares commonalities with the job-shop scheduling [10]. The problem is to find the job sequences on machines to achieve a goal (e.g., minimize makespans), while the machine sequence of the jobs is fixed [11]. There has been considerable research in the area of job shop scheduling over the past years [12]. As it is NP-hard [13] and one of the most computationally intractable combinatorial problems [14], heuristic techniques have been developed such as dispatching rules, shifting bottleneck heuristic, and local search [12]. Among those techniques, dispatching rules receive massive attention from a practical viewpoint [15]. A dispatching rule is used to assign a job to a resource at a given time when a resource becomes available for the operation. This approach is useful to find a reasonably good schedule concerning an objective such as the makespan, the total completion time, or the maximum lateness in a relatively short time.

A dispatching rule is only applicable when we are aware of required parameters such as the release time, the processing time, the sequence of operations of jobs. In many circumstances, however, we have limited information about the scheduling parameters [16]. Imagining an emergency department of a hospital, we do not know when and why a patient would come into the department before the visit happens. Furthermore, there can be irregular clinical procedures even for the patients diagnosed with the same disease since exceptions are always able to occur. Even worse is that we are unaware of the processing time taken to finish an operation, making it difficult to assign the most efficient resource to patients. This problem is called a non-clairvoyant online-over-time problem [17] where the decision-maker do not know the data concerning a job when the job is presented to him. In this problem, we cannot apply the developed dispatching rules to

978-1-7281-0919-0/19/$31.00 ©2019 IEEE
DOI 10.1109/ICPM.2019.00027

make an optimal decision.

A natural idea to deal with the above example is first to predict relevant parameters and then utilize the predictions to optimize the resource scheduling. To achieve this goal, two challenges need to be addressed. (i) How to effectively build prediction models to generate the required parameters? (ii) How to efficiently dispatch resources based on the predictions? Thus, we propose a two-phase method to optimize resource allocation based on prediction to efficiently solve the non-clairvoyant online problem under the objective of minimizing total weighted completion time.

In this paper, we demonstrate how the results from predictive business process monitoring can be transformed into a concrete process improvement action. To this end, we solve a non-clairvoyant online-over-time problem with the two-phase method integrating predictions with resource allocation optimization. To the best of our knowledge, this is the first work to solve the problem by associating results from predictive business process monitoring. To verify the effectiveness and efficiency of our proposed method, we evaluate it on both an artificial and a real-life event log.

In the rest of this paper, we explain backgrounds in Section II and provide a running example, a baseline approach and insights to solve the problem in Section III. In Section IV, we detail the two-phase method, i.e., offline prediction model construction and online resource scheduling. Section V presents how we evaluate our suggested method both on artificial and real-life event logs. Then we review related work in Section VI and concludes this paper in Section VII.

II. Backgrounds

This section presents preliminaries and notations that will be required in the remainder of this paper. We also elaborate non-clairvoyant online job shop scheduling problem we try to solve in this paper.

A. Preliminaries

Let $\mathcal{E}$ be the event universe. Each event is characterized by its attributes such as activity, resource, timestamp.

Definition 1 (Trace, Event Log)**.** *A trace $\sigma = e_1,e_2,...,e_n \in \mathcal{E}^*$ such that each event occurs only once, i.e., for $1 \leq i < j \leq |\sigma| : e_i \neq e_j$. An event log $\mathcal{L}$ is a multi-set of traces such that each event appears at most once in the entire log.*

Definition 2 (Prefix, Suffix)**.** *A prefix of length k ($0 < k < n$) of a trace $\sigma = \langle e_1, e_2, ..., e_n \rangle$ is $h_k(\sigma) = \langle e_1, e_2, ..., e_k \rangle$ and its suffix is $t_k(\sigma) = \langle e_{k+1}, ..., e_n \rangle$. For instance, for $\sigma_i = \langle e_1, e_2, e_3, e_4, e_5 \rangle$, $h_3(\sigma_i) = \langle e_1, e_2, e_3 \rangle$ and $t_3(\sigma_i) = \langle e_4, e_5 \rangle$.*

Let $\mathcal{A}, \mathcal{R}$ and $\mathcal{T}$ be the set of activities, resources, and processing times, respectively.

Definition 3 (Event representation function)**.** *A function $\pi_{\mathcal{A}} \in \mathcal{E} \rightarrow \mathcal{A}$ assigns process activities to each event, and a function $\pi_{\mathcal{R}} \in \mathcal{E} \rightarrow \mathcal{R}$ assigns to each event a resource. A function $\pi_{\mathcal{T}} \in \mathcal{E} \rightarrow \mathcal{T}$ assigns processing times to events.*

Table I lists the notations used throughout the paper.

TABLE I
Summary of Symbol Notations

Notation	Description
I, R, A	Set of instances, resources, and activities
WI	Set of work items
$wi_{i,k}$	k_{th} work item of instance I_i
$p_{i,k,j}$	Processing time of work item $w_{i,k}$ by R_j
ri_i, rr_j	Remaining time for I_i, R_j to be ready
S_i	Start time of I_i
F_i	Finish time of I_i
C_i	Completion time of I_i ($C_i = F_i - S_i$)
w_i	Weight of I_i

B. Problem Statement

In this subsection, we define non-clairvoyant online job shop scheduling problem which we endeavor to solve using the two-phase method.

Definition 4 (Non-clairvoyant Online Job Shop Scheduling)**.** *Given a set of instances I, where each instance has a set of operations which needs to be processed in a specific order, non-clairvoyant online job shop scheduling problem finds an optimal scheduling of all operations within instances while minimizing $\sum_i w_i C_i$, where w_i is the weight of I_i and C_i is the difference between the finish time, F_i, and start time, S_i, of I_i. We also make some assumptions as follows.*

- *We are unaware of the data concerning an instance except for the weight of it.*
- *We only find out what the next operation of an instance is when an instance finishes its current operation.*
- *Each operation has a specific set of resources that it needs to be processed on and only one operation within an instance can be processed at a given time.*
- *An operation cannot be preempted, so once processing begins on an operation, it cannot be stopped until complete.*

III. Baseline Approach and Insights

In the following, we describe a running example and a baseline approach called WeightGreedy. Also, we explain the insights from which we develop our suggested method.

A. Running Example

Throughout this paper, a simple situation described in Fig. 1-(c) will serve as a running example. As shown in Fig. 1-(a), a node on the left means a work item, while a node on the right indicates a resource. An arc between nodes represents that a work item can be processed by a resource and its label means the processing time taken for the resource to finish the work item. Weights of instances are listed in Fig. 1-(b). Assume we are now at $T = t$. There are four work items (i.e., $wi_{1,1}(1_{st}$ work item of instance $I_1), wi_{2,2}, wi_{3,1}, wi_{4,1})$ and three resources (i.e., r_1, r_2, r_3) available at the moment. When allocating resource in this moment, we are unaware of a red value on an arc, (i.e., the processing time of a work item by a resource), a green value under a node (i.e., the start time

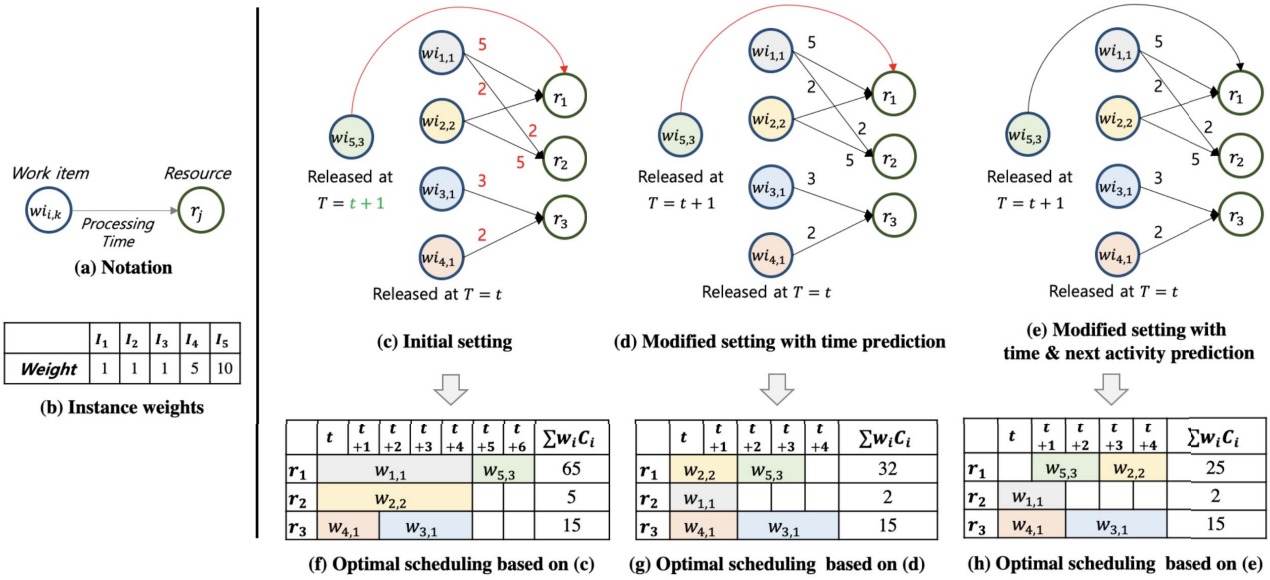


Fig. 1. An example of different problem settings and optimal schedules

of a new work item), and a red line between the left and right nodes (i.e., the resource requirement of a work item).

B. A Baseline Approach

It is non-trivial to optimally assign resources in the above example since we only have limited information about the process. In this subsection, we first introduce a baseline solution called WeightGreedy. The main idea of WeightGreedy is that each work item is assigned to an available resource in a "first come, first served" manner. If there exist conflicting demands for the same resource, the work item with higher instance weight is served first. If the competing work items have the same instance weights, the tie is broken at random.

Fig. 1-(f) shows the optimal resource scheduling based on the baseline approach. Note that the scheduling is conducted under pull mechanism, i.e., work items are offered to available resources which can freely pick any of them [18]. There exist four work items, which are released at $T = t$ as described in Fig. 1-(c). Since both $wi_{1,1}$ and $wi_{2,2}$ requires r_1 and they have the same instance weight (i.e., $w_1 = w_2 = 1$), we randomly assign $wi_{1,1}$ to r_1. After that, $wi_{2,2}$ is assigned to r_2. Next, r_3, which has two demands, is allocated to $wi_{4,1}$ since w_4 (i.e., 5) is higher than w_3 (i.e., 1). At $T = t+1$, $wi_{5,3}$ has just released, and $wi_{3,1}$ remains in the waiting list since it was not assigned at $T = t$. As there are no resources available at the moment, those work items are staying in the waiting list. Consequently, $wi_{3,1}$ is processed at $T = t + 2$ and $wi_{5,3}$ at $T = t + 5$, as shown in Fig. 1-(f). The subtotal weighted completion times for resources are 65, 5, 15, resulting in a total sum of 85.

C. Insights

First insight we find out is that we can improve the policy regarding the conflicting demands of a resource by predicting the processing times of running work items. As shown in Fig. 1-(d), it is better to assign $wi_{1,1}$ to r_2 and $wi_{2,2}$ to r_1 since these matches have lower processing time than the ones from the baseline approach, which then results in lower total weighted completion time. The optimal scheduling is described in Fig. 1-(g). The total weighted completion time is reduced to 59 from 85 due to the time predictions.

Secondly, we identify that we can further improve resource scheduling if we reserve resources for instances that have higher weights. As shown in Fig. 1-(e), if we know, in advance, that r_1 is required by $wi_{5,3}$ at $T = t+1$, we can reserve r_1 to serve it since I_5 has much higher weight than the others. Thus, our optimal solution, at $T = t$, is to make pseudo-assignment of $wi_{5,3}$ to r_1 and execute it right after $wi_{5,3}$ is released, i.e., at $T = t+1$. To figure out that I_5 requires r_1, we need first to predict the next activity of I_5 and then to find resources that can serve the activity, i.e., r_1 in this case. Resulting schedule is shown in Fig. 1-(h), where the total weight completion time of r_1 decreases from 32 to 25.

IV. Method

This section proposes a two-phase method, which optimizes resource scheduling based on the time and next event prediction in the non-clairvoyant online setting. A general overview is presented first and, afterward, we explain each step in more detail.

A. Overview

Our method consists of two phases: *offline prediction model construction* and *online resource scheduling*. Fig. 2 describes the overview of this method.

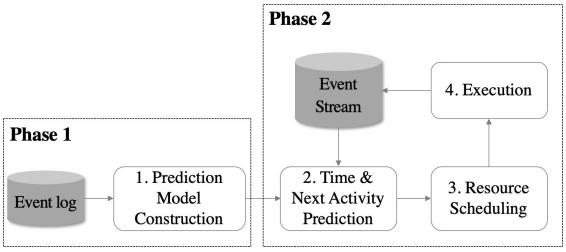


Fig. 2. Overview of two-phase method

1) Phase 1: Offline Prediction Model Construction: The first phase of our suggested method is to build prediction models both on the processing time and the next activity of ongoing instances based on an event log. It has been actively studied to predict the remaining time and the next activity of ongoing instances [6], [7], [19]. In this paper, we use one of the state-of-the-art methods suggested in [19], which deploys LSTM neural networks [20]. The model is trained on one-hot encoded vectors of historical traces which are obtained by transforming data from the event log.

2) Online Resource Scheduling: The second phase consists of three steps: *time and next event prediction*, *resource scheduling*, and *execution*. In this phase, optimal resource allocation takes place, based on the predictions on running instances whose information is recorded in event stream. First, we predict the processing time and the next activity of each running instance. For the predictions, we use the prediction model built in phase 1. Next, they are passed into the resource scheduling step, where a pseudo-assignment between a work item and a resource is established. Afterward, the pseudo-assignment is executed for any valid pairs of a work item and a resource. The event stream is updated every time the execution is carried out and this becomes an input for the next iteration of these three steps.

B. Prediction Model Construction

In this step, we aim at building a time prediction function f_t and a next activity prediction function f_a such that $f_t(hd^k(\sigma)) = hd^1(tl^{k-1}(\pi_{\mathcal{T}}(\sigma)))$ and $f_a(hd^k(\sigma)) = hd^1(tl^k(\pi_{\mathcal{A}}(\sigma)))$, given k. Fig. 3 describes an architecture of our prediction model, which has two shared LSTM layers followed by two specialized layers for each task, i.e., time prediction and next activity prediction. An input is produced from an event log by transforming historical event $e \in hd^k(\sigma)$ with one-hot-encoding. In other words, each event $e_i \in hd^k(\sigma)$ is encoded as a vector of length $|A| + |R|$, such that $|A| + |R|$ features are all set to 0 except the one occurring at the index of $\pi_{\mathcal{A}}(e_i)$ and $\pi_{\mathcal{R}}(e_i)$.

Two target outputs of the model, i.e., processing time and next activity, are generated from the event log as well. The

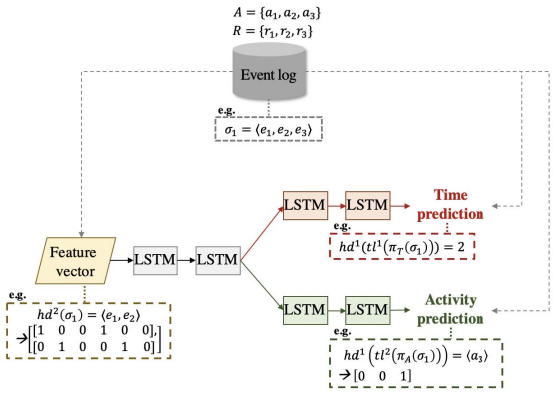


Fig. 3. An architecture of prediction model and a training example

first output, processing time, is a numerical value defined as $hd^1(tl^{k-1}(\pi_{\mathcal{T}}(\sigma)))$. The second output, next activity, is a categorical value represented as $hd^1(tl^k(\pi_{\mathcal{A}}(\sigma)))$. It is then encoded with one-hot-encoding as we do when generating a feature vector.

Suppose we train our model with $\sigma_1 = \{e_1, e_2, e_3\}$ and $k = 2$, shown in Fig. 3. The first two events of σ_1, i.e., $hd^2(\sigma_1) = \{e_1, e_2\}$, become a training input of out model after transforming them into a one-hot-encoded vector. Since we have $A = \{a_1, a_2, a_3\}$ and $R = \{r_1, r_2, r_3\}$, the length of the one-hot-encoded vector for each event, e_1 and e_2, is set to 6. As $\pi_{\mathcal{A}}(e_1) = a_1$, and a resource, $\pi_{\mathcal{R}}(e_1) = r_2$, the one-hot-encoded vector becomes $[1, 0, 0, 1, 0, 0]$. We can calculate the one-hot-encoded vector for e_2 in the same manner, i.e., $[0, 1, 0, 0, 1, 0]$. Our first target output is $hd^1(tl^1(\pi_{\mathcal{T}}(\sigma_1)))$, i.e., the processing time of e_2, 2. The second one is $hd^1(tl^2(\pi_{\mathcal{A}}(\sigma_1)))$, i.e., the activity of e_3, a_3, which is then transformed to $[0, 0, 1]$.

We train all sets of network weights using *Adam algorithm* [21] such that the mean absolute error (MAE) between the actual processing time and the predicted processing time, and the cross-entropy between the actual next activity and predicted next activity are minimized. All weights are initialized with *Xavier Initialization* [22]. We use 100 dimensions for the size of the LSTM memory. As regularization strategies, we use *Dropout* [23] and *Batch Normalization* [24].

C. Time & Next Activity Prediction

Based on the prediction model we construct in the previous step, we predict the processing time and the next activity of ongoing instances from the event stream. Event stream provides information of running instances. Whenever a work item of an instance is initialized, we conduct two consecutive predictions regarding the instance. The first prediction aims at predicting the next operation of the instance. Based on this next activity prediction, we create artificial events ($\widehat{e}$) by combining the predicted next activity with available resources. Each of these artificial events, coupled with historical events, becomes an input for the second prediction. The purpose of

this prediction is to predict the processing time of the artificial event. Note that we iterate this second prediction for each artificial event. These prediction results are used to enhance the network as shown in Fig. 1-(e). The results are updated when the running instance is ready for the next operation only if the previous prediction result on next activity differs from the actual one.

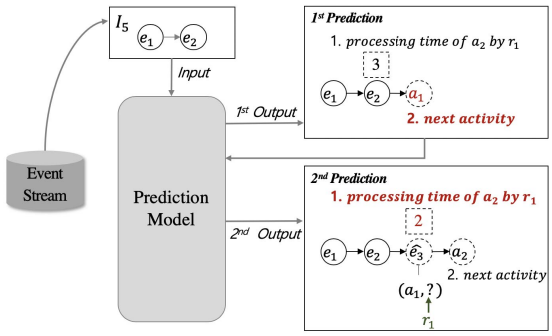


Fig. 4. An example of time and next activity prediction

For I_5 in Fig. 4, we conduct the first prediction based on the records from event stream. From the one-hot-encoded vector of $\{e_1, e_2\}$, the processing time of e_2 and the next activity a_1 are predicted. The next activity prediction result, a_1, is used to constitute an artificial event $\hat{e_3}$ with r_1 who is the only available resource for a_1. The predicted processing time of a_1 by r_1 is 2. These prediction results enable us to consider I_5 for optimal resource allocation at $T = t$ as described in Fig. 1-(e).

D. Resource Scheduling

In this step, we solve a min-cost max-flow network problem, which aims at finding a maximum flow with the smallest possible cost, in order to optimize resource allocation. We start by constructing a bipartite graph such that nodes on the left are work items and those on the right are resources. Using the results from the previous step, we can include not only the work items and the resources in the queue but also those in process. For example, in Fig. 5, the graph includes a dotted circle, which represents a work item which is expected to appear later. The set of nodes on the left (right) is denoted by $\widehat{WI}$ ($\widehat{R}$). Afterward, we add an edge between a pair of nodes in the bipartite graph, i.e., (wi, r) ($wi \in \widehat{WI}$, $r \in \widehat{R}$). Each edge contains $(cost, capacity)$, where $cost = (p_{i,k,j} + max(ri_i, rr_j, 0))/w_i$ such that $p_{i,k,j}$ is the processing time of work item $w_{i,k}$ by r_j, ri_i is the remaining time for I_i to be ready, and rr_j is the remaining time for r_j. Note that we devise the cost function to optimize our objective (i.e., minimizing the total weighted completion time) by assigning less cost to edges which have less processing time and higher instance weight, while giving a penalty if a work item or a resource is not prepared. Finally, we adopt the minimum cost maximum flow algorithm based on network simplex method [25] to generate an optimal matching between work items and resources, i.e., pseudo-assignment.

Algorithm 1 illustrates the generation of a pseudo-assignment between work items and resources. Both predicted and prepared work items are instantiated to the set of nodes on the left, $\widehat{WI}$, while predicted and prepared resources are instantiated to the set of nodes on the right, $\widehat{R}$, in a bipartite graph. In lines 1-7, we create a source node and a sink node and then we add edges connecting source node to left nodes and right nodes to sink node. Each edge has a cost of zero and a capacity of one. In lines 8-12, we add edges between left nodes and right nodes if the work items (left) can be processed by resources (right), where the cost is calculated as $(p_{i,k,j} + max(ri_i, rr_j, 0))/w_i$ and the capacity is one.

Algorithm 1 Resource scheduling algorithm

Input: $\widehat{WI}, \widehat{R}$
Output: Pseudo-Assignment $\widehat{M}$
1: *Produce source node s, sink node t*;
2: **for** *node* $wi_{i,k} \in \widehat{WI}$ **do**
3: *add edge* $(s, wi_{i,k}, (0, 1))$
4: **end for**
5: **for** *node* $r_j \in \widehat{R}$ **do**
6: *add edge* $(r_j, t, (0, 1))$
7: **end for**
8: **for** *node* $wi_{i,k} \in \widehat{WI}$ **do**
9: **for** *node* $r_j \in \widehat{R}$ **do**
10: $c \leftarrow (p_{i,k,j} + max(ri_i, rr_j, 0))/w_i$
11: *add edge* $(wi_{i,k}, r_j, (c, 1))$
12: **end for**
13: **end for**
14: $\widehat{M} \leftarrow MinCostMaxFlow(s, t)$
15: **return** $\widehat{M}$

In Fig. 5, $\widehat{WI}$ has five elements, one of which is in progress and four of which are ready for the assignment and all three elements of $\widehat{R}$ are ready for assignments. After running a min-cost max-flow algorithm, we have a pseudo-assignment of three matches represented as bold edges, i.e., $w_{5,3}$ *to* r_1, $w_{1,1}$ *to* r_2, $w_{4,1}$ *to* r_3.

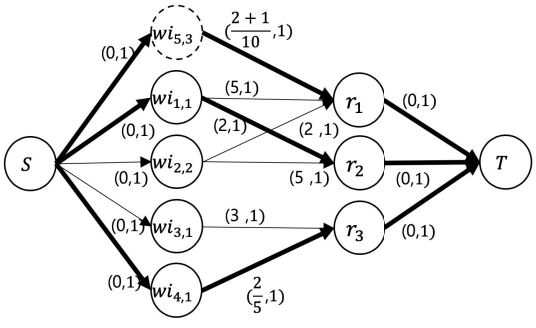


Fig. 5. An example of resource scheduling

E. Execution

In this step, we explain how the pseudo-assignment is executed. The main idea is that we filter executable matches from the pseudo-assignment and execute them. An executable match means both corresponding instance and resource are available at the current time, while a non-executable match indicates that either of them is not available.

In Fig. 6-(a), the solid circle represents availability of an instance or resource, while the dotted circle means unavailability. Thus, there exist two executable matches, i.e., blue arcs, and one non-executable match, i.e., red arc. We filter only executable matches as shown in Fig. 6-(b) and execute those matches. In other words, $wi_{1,1}$ and $wi_{4,1}$ are processed by r_2 and r_3, respectively.

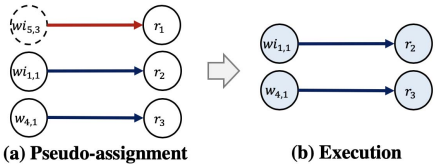


Fig. 6. An example of execution

V. Evaluation

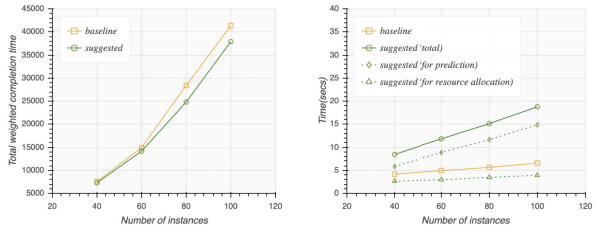
A twofold approach is used to evaluate the proposed method. First, we evaluate the ability of our proposed method to optimize resource allocation in the non-clairvoyant online setting using an artificial event log. Second, we discuss the application of the method on a real-life log.

A. Artificial event log

1) Experimental design: Our proposed method is evaluated by comparing its ability to solve a non-clairvoyant online job shop problem with the baseline approach in terms of total weighted completion time. To this end, an artificial log is generated by simulating a simplified process at an emergency department of a hospital, composed of 11 activities and 25 resources. The department operates 24 hours a day. Each resource has his/her own set of activities that they are able to serve and the processing times of work items vary depending on the proficiency level. Patients with different weights come into the process in a regular interval. We assume a non-clairvoyant online environment, so we do not know how long the current operation of a patient will take before it finishes. In addition, we find out what the next operation of a patient will be only after it finishes its current operation.

We simulate this process for 7 days to produce an event log which will be used to construct a prediction model. For each activity, we define processing times of its available resources, which are assumed to follow *Gaussian* distribution. Each patient is assigned a weight ranging from 1 to 10 which is assumed to follow a uniform distribution. The resulting log has 1000 patients and 25 resources. We build an event stream by generating 100 patients entering the department in a regular interval for 6 hours. Each patient has his/her future activities which are unknown when making a schedule.

2) Results: Based on the event stream we generated, we compare our proposed method with the baseline, Weight-Greedy, in terms of total weighted completion time. Fig. 7 shows experimental results on varying number of instances(i.e., patients). The total weighted completion time increases with the increase of $|I|$ for both WeightGreedy and our suggested method Fig. 7-(a). The proposed method outperforms the baseline approach in all sizes of instances since it takes potential work items into account. For example, when the number of instances is 80, the total weighted completion time of the baseline approach (i.e., 28,393) is 14 percent higher than the one of the suggested method (i.e., 24,804). Both algorithms need more time when $|I|$ increases as shown in Fig. 7-(b). The computation time for the suggested method is relatively higher than the baseline approach, due to its prediction step which occupies most of the computations.



(a) Total weighted completion time of varying |I| (b) Computation time of varying |I|

Fig. 7. Experimental results on varying $|I|$

B. Real-life event log

1) Experimental design: For the evaluation of our proposed method on real-life problem, we use an event log from Business Processing Intelligence Challenge 2012 (BPIC'12). It contains the data regarding the application procedure for a personal loan or overdraft at a global financing organization over a roughly six month period from October 2011 to March 2012. The dataset is comprised of a total of 262,200 events within 13,087 cases, recording the procedure from submitting an application to receiving a conclusion such as approval, cancellation, and rejection. This process is classified into three major types: one that refers to the states of the application itself, one that refers to the states of an offer, and one that tracks the states of work items that occur during the approval process. Among the three types, we investigate the third one containing events which are executed manually since we are not interested in the events performed automatically. Each case contains a case attribute, AMOUNT_REQ, which represents the amount requested in the application. Based on it, we create 10 equally spaced buckets, such that the bucket having a label of 1 contains the cases with the lowest amount and the bucket having a label of 10 includes the cases with the highest amount. Each case is given a weight according to the label of a bucket where it belongs.

We build an event log from the dataset by filtering it to include the events before 10/3/2012 and use it as an input of phase 1 in our method. The instances and resources, who appear on 10/3/2012 in the dataset, are extracted from the dataset, and they become components of the event stream which we use in phase 2. For those instances, we only consider operations that took place on 10/3/2012, to produce work items. The information, such as the activities and resources an instance has experienced before 10/3/2012, is also used to make predictions. The resulting number of instances is 110, and each instance has conducted 3 activities for the date on average.

2) Results: From the prediction model and the event stream, we simulate the resource allocation on 10/3/2012. Table II shows the results from both baseline approach and our proposed method. The total weighted completion time of baseline approach (i.e., 1,479) is 42 percent higher than the one of our suggested method (i.e., 1038). The massive difference comes from assigning the most efficient resource to the work item and reserving resources to enable them process instances having higher weights. The computation time is much higher in the proposed method. This is because there are many arcs between instances and resources, i.e., each work item has many resource options, which requires high computation when predicting the parameters (i.e., 110.1 secs out of 115.6 secs).

TABLE II
EXPERIMENTAL RESULTS ON REAL-LIFE EVENT LOG

Method	Total weighted completion time	Computation time(secs)
Baseline	1479	7.6
Suggested	1038	115.6

VI. RELATED WORK

Our work is related to the research on predictive business process monitoring and job shop scheduling.

A. Predictive Business Process Monitoring

The prediction tasks can mainly be classified into four different categories: time, risk probability, performance indicators, next event [5]. Among them, time and next event prediction provide valuable inputs for resource allocation [19].

1) Time Prediction: Most of the studies to predict time-related values focus on predicting the remaining time of running cases [5]. The first framework for this problem is suggested in [6] and many approaches have been developed based on it. Folino et al. [26] extends the technique described in [6] by clustering the log traces according to the corresponding context features. Polato et al. [7] further enhance the approach by adding machine learning model on additional attributes of events. These approaches, however, assume that the underlying process is stationary, which is not always true [7]. To mitigate this limitation, Tax et al. [19] developed a method to predict both time and next event with Long Short Term Memory network, where there is no need for an explicit representation of the process.

2) Next Event Prediction: Relatively few works have been done in next event prediction. Most works use an explicit process model representation such as Hidden Markov Model (HMM). Le et al. [27] proposes hybrid Markov models for predicting the next step in a process instance. If an instance reaches an unknown state, the model results in the prediction based on the most similar state by applying edit distance. Lakshmanan et al. [28] suggests a method that uses the instance-specific Probabilistic Process Models (PPM) where state transition probabilities for a Markov chain is derived from decision trees mined from case attributes. Recently, Evermann et al. [29] proposes a method based on LSTM neural networks with embedding as an encoding technique. Tax et al. [19] suggests an LSTM-based model with one-hot-encoding and multi-task learning to improve the prediction accuracy.

B. Job Shop Scheduling

A job shop scheduling problem consists of a set of machines that perform operations on jobs, where each job has a processing order through the machines. There are several constraints on jobs and machines. While the machine sequence of the jobs is fixed, the problem is to find the job sequences on the machines which optimizes an objective (e.g. minimize the makespan) [12]. It is well known that the problem is NP-hard [13] and belongs to the most intractable problems [14].

A huge amount of literature on the scheduling of job shops has been published in the last decades [12]. The approaches for solving this problem can be categorized into three groups: dispatching, shifting bottleneck heuristic, and local search [12]. However, the application of these approaches to practical usages is somewhat limited because most of these techniques are not amenable to actual utilization in real job shops [30]. Dispatching rules, on the other hand, have been widely adopted in industry as they are computationally efficient, and robust to uncertainty [15].

More than 100 dispatching rules are suggested in the literature [31]. However, there's no rule which is applicable in non-clairvoyant online job shop problem where we do not know basic information for dispatching [16]. To the best of our knowledge, there is no attempt to utilize the prediction results to improve the applicability and efficiency of dispatching rules.

VII. CONCLUSION

In this paper, we suggest a concrete method to improve a business process using results from predictive business process monitoring. We start by proposing a problem of online resource allocation in non-clairvoyant online environment. To address the problem, the key insight is to predict future behaviors of instances and resources in the process. Based on the insight, we devise a novel two-phase method, which integrates offline prediction model construction with online resource scheduling. For the prediction model, we adopt one of the state-of-the-art techniques based on LSTM neural networks to predict both the processing time and the next activity of a running instance. For online resource scheduling, we utilize the predictions generated from event stream to build a bipartite

graph, after which we solve a minimum cost and maximum flow problem. We verify the effectiveness and efficiency of the proposed method on both an artificial log and a log from BPIC'12. We anticipate our work as a leap on the road to more intelligent prediction-based process improvement techniques for a wide range of domains.

The proposed method has several limitations. First, our proposed method relies heavily on the performance of the prediction model. If a prediction for the processing time of a work item differs, we fail to match the work item to the most efficient resource. Second, the computation time is relatively higher than the baseline approach. The high computation comes from the execution of predictions for every newly-assigned instances, after which the search space for solving a network problem increases.

For future work, we will extend this two-phase method to achieve another goal such as minimizing the potential risks in the business process by predicting other relevant parameters and defining a relevant cost function of network arcs. Another direction for future work is to extend the proposed method by adopting advanced dispatching techniques.

ACKNOWLEDGMENT

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2018-0-0144) supervised by the IITP (Institute for Information & communications Technology Promotion)

REFERENCES

[1] W. M. P. van der Aalst, *Process Mining: Data Science in Action*, 2nd ed. Heidelberg: Springer, 2016.

[2] W. M. P. van der Aalst, M. L. Rosa, and F. M. Santoro, "Business process management - don't forget to improve the process!" *Business & Information Systems Engineering*, vol. 58, pp. 1–6, 2016.

[3] A. Metzger, P. Leitner, D. Ivanovi, E. Schmieders, R. Franklin, M. Carro, S. Dustdar, and K. Pohl, "Comparing and combining predictive business process monitoring techniques," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 2, pp. 276–290, Feb 2015.

[4] M. de Leoni, W. M. P. van der Aalst, and M. Dees, "A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs," *Information Systems*, vol. 56, pp. 235–257, 2016.

[5] A. E. Mrquez-Chamorro, M. Resinas, and A. Ruiz-Cortes, "Predictive monitoring of business processes: A survey," *IEEE Transactions on Services Computing*, vol. 11, no. 6, pp. 962–977, Nov 2018.

[6] W. M. P. van der Aalst, M. Schonenberg, and M. Song, "Time prediction based on process mining," *Information Systems*, vol. 36, no. 2, pp. 450–475, 2011.

[7] M. Polato, A. Sperduti, A. Burattin, and M. D. Leoni, "Time and activity sequence prediction of business process instances," *Computing*, vol. 100, no. 9, pp. 1005–1031, Sep 2018.

[8] Z. Huang, W. M. P. van der Aalst, X. Lu, and H. Duan, "Reinforcement learning based resource allocation in business process management," *Data & Knowledge Engineering*, vol. 70, no. 1, pp. 127–145, 2011.

[9] W. Zhao and X. Zhao, "Process mining from the organizational perspective," in *Foundations of Intelligent Systems*, Z. Wen and T. Li, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 701–708.

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

[11] D. Applegate and W. J. Cook, "A computational study of the job-shop scheduling problem," *INFORMS Journal on Computing*, vol. 3, pp. 149–156, May 1991.

[12] J. Baewicz, W. Domschke, and E. Pesch, "The job shop scheduling problem: Conventional and new solution techniques," *European Journal of Operational Research*, vol. 93, no. 1, pp. 1–33, 1996.

[13] R. Graham, E. Lawler, J. Lenstra, and A. Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," in *Discrete Optimization II*, ser. Annals of Discrete Mathematics, P. Hammer, E. Johnson, and B. Korte, Eds. Elsevier, 1979, vol. 5, pp. 287–326.

[14] E. L. Lawler, J. K. Lenstra, A. H. R. Kan, and D. B. Shmoys, "Chapter 9 sequencing and scheduling: Algorithms and complexity," in *Logistics of Production and Inventory*, ser. Handbooks in Operations Research and Management Science. Elsevier, 1993, vol. 4, pp. 445–522.

[15] C. Tsiushuang, "Dispatching rules for manufacturing job-shop operations," *IFAC Proceedings Volumes*, vol. 26, no. 2, Part 4, pp. 975–978, 1993, 12th Triennal Wold Congress of the International Federation of Automatic control. Volume 4 Applications II, Sydney, Australia, 18-23 Jul.

[16] N. Megow, "Coping with incomplete information in scheduling — stochastic and online models," in *Operations Research Proceedings 2007*, J. Kalcsics and S. Nickel, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 17–22.

[17] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 3rd ed. Springer Publishing Company, Incorporated, 2008.

[18] A. Kumar, W. M. P. van der Aalst, and E. M. W. Verbeek, "Dynamic work distribution in workflow management systems: How to balance quality and performance," *Journal of Management Information Systems*, vol. 18, no. 3, pp. 157–193, Jan. 2002.

[19] N. Tax, I. Verenich, M. La Rosa, and M. Dumas, "Predictive business process monitoring with lstm neural networks," in *Advanced Information Systems Engineering*, E. Dubois and K. Pohl, Eds. Cham: Springer International Publishing, 2017, pp. 477–492.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–1780, Dec 1997.

[21] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, Dec 2014.

[22] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *Journal of Machine Learning Research - Proceedings Track*, vol. 9, pp. 249–256, Jan 2010.

[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, Jun 2014.

[24] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15. JMLR.org, 2015, pp. 448–456.

[25] W. H. Cunningham, "A network simplex method," *Mathematical Programming*, vol. 11, no. 1, pp. 105–116, Dec 1976.

[26] 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, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 287–304.

[27] M. Le, B. Gabrys, and D. Nauck, "A hybrid model for business process event prediction," in *Research and Development in Intelligent Systems XXIX*, M. Bramer and M. Petridis, Eds. London: Springer London, 2012, pp. 179–192.

[28] G. T. Lakshmanan, D. Shamsi, Y. N. Doganata, M. Unuvar, and R. Khalaf, "A markov prediction model for data-driven semi-structured business processes," *Knowledge and Information Systems*, vol. 42, no. 1, pp. 97–126, Jan 2015.

[29] J. Evermann, J.-R. Rehse, and P. Fettke, "A deep learning approach for predicting process behaviour at runtime," in *Business Process Management Workshops*, M. Dumas and M. Fantinato, Eds. Cham: Springer International Publishing, 2017, pp. 327–338.

[30] X. Qiu and H. Y. Lau, "An ais-based hybrid algorithm with pdrs for multi-objective dynamic online job shop scheduling problem," *Applied Soft Computing*, vol. 13, no. 3, pp. 1340–1351, 2013, hybrid evolutionary systems for manufacturing processes.

[31] Y. L. Chang, T. Sueyoshi, and R. S. Sullivan, "Ranking dispatching rules by data envelopment analysis in a job shop environment," *IIE Transactions*, vol. 28, no. 8, pp. 631–642, 1996.