# A framework for encoding the multi-location load state of a business process

Björn Rafn Gunnarsson[1], Jochen De Weerdt[1] and Seppe vanden Broucke[2,1]

[1]*Research Center for Information Systems Engineering (LIRIS), KU Leuven, Naamsestraat 69, Leuven 3000, Belgium*
[2]*Department of Business Informatics and Operations Management, UGent, Tweekerkenstraat 2, Ghent 9000, Belgium*

**Abstract**
Predictive process monitoring techniques proposed in the literature have in general relied on intra-case features in order to make predictions. Consequently, these techniques do not incorporate inter-case information, e.g. reflecting the overall state of the process. In this work, a novel framework for encoding inter-case features is proposed. This framework enriches events with a representation of the multi-location load state of a business process. The framework solely relies on deriving the load state at relevant locations in the process and therefore provides users with a straightforward approach for incorporating the status of a business process. The framework was evaluated using a typical predictive process monitoring setup over a number of real-life event logs where predictive process monitoring techniques were developed for predicting the remaining processing time of ongoing cases. A general performance gain was observed for models that use inter-case features encoded using the proposed multi-location load state framework compared to models that only consider intra-case features in order to make predictions. This opens up interesting avenues for future research including expanding the proposed framework so that it additionally incorporates forecasts for the future multi-location load state of a process under investigation.

**Keywords**
Predictive process monitoring, Inter-case features, Process mining, Remaining time prediction

## 1. Introduction

Modern businesses rely on process-aware information systems to support their business processes. Consequently, detailed information can be extracted from these systems in the form of event logs containing information on the actual execution of the business processes [1]. Predictive process monitoring (PPM) is a domain of research which concerns itself with developing predictive models based on historical process executions [2]. These methods can provide organizations with valuable insights in order to improve business operations, e.g. by providing predictions for future process violations and/or delays for ongoing cases. This in turn allows businesses to carry out preventive measures in order to mitigate the loss resulting from such deviations [3, 4]. Consequently, a variety of predictive process monitoring techniques have been proposed for different prediction tasks such as remaining time prediction [5, 6, 7], outcome-oriented prediction [8, 9, 10], next event prediction [11, 12, 13] and trace prediction [14, 15, 16].

*PMAI@IJCAI22: International IJCAI Workshop on Process Management in the AI era, July 23, 2022, Vienna, Austria*
bjornrafn.gunnarsson@kuleuven.be (B. R. Gunnarsson); jochen.deweerdt@kuleuven.be (J. De Weerdt); seppe.vandenbroucke@ugent.be (S. vanden Broucke)

In general, PPM techniques proposed in the literature rely on encoding schemes where intra-case features, i.e. providing information about the execution history of a specific ongoing case of interest, are constructed and utilized in order to make predictions. Therefore, these techniques assume that the processing of a case solely depends on the attributes of the case itself. However, cases are not processed in isolation and their processing can potentially be influenced by other cases processed in the system [17]. Recently, techniques have been proposed in order to extract relevant inter-case features from event logs [18, 19, 20]. The motivation for this development is to allow PPM techniques to additionally incorporate information on the overall status of the process under investigation in order to make improved predictions for an ongoing case of interest.

In this work, we propose a framework for generating inter-case features enriching events with information on the multi-location load state of the process. A pivotal advantage of this framework is that it solely relies on encoding the load state at relevant locations in the process where locations are defined based on control-flow information. It therefore provides practitioners with a straightforward approach for incorporating the relevant status of a business process in order to make improved predictions when developing PPM techniques. Two configurations of the suggested framework are considered. On the one hand, a general system based encoding is proposed, containing a representation of the load state at a fixed set of important locations in the system. On the other hand, a case based encoding is introduced, providing a representation of the status of a process in close proximity to a case of interest. Additionally, two approaches for deriving the load state for a single location in a business process are considered and compared. The first counts the number of active cases in a location and therefore provides a snapshot view of the current status at a given location. The second identifies an optimal time window and counts the number of cases which have been processed at the location during this optimal time window.

The remainder of this paper is structured as follows. An overview of the relevant literature is provided in Section 2. The following section provides relevant background information. The framework for encoding the multi-location load state of a business process is introduced and discussed in Section 4. Section 5 provides details on the considered experimental setup and the results of the experimental evaluation of the proposed framework. This is followed by a more detailed discussion on the obtained results in Section 6. The last section of this paper provides concluding remarks and opportunities for future work.

## 2. Literature review

The development of accurate PPM methods has been extensively studied in the current literature. However, the lion's share of research in PPM has solely considered intra-case information, i.e. information originating from the execution history of the case of interest, in order to make predictions for that case. These studies therefore assume that cases are processed in isolation and therefore are not affected by the processing of other cases. However, this assumption seems unrealistic for many processes, e.g. where cases compete for a limited number of resources [18, 21].

Nonetheless, a few papers have incorporated a limited number of inter-case features when developing PPM techniques. In [8], the authors considered the total number of active cases at the time of execution of a given event as a feature when predicting the outcome of cases. Similarly, [22] considered the number of active cases when predicting the remaining processing time of cases. Additionally, a few studies (see e.g. [20, 23, 24]) have considered using the so called performance spectrum [17] in order to extract a number of performance related inter-case features which in turn can be utilized for predictive performance monitoring tasks.

Previous research most related our work is proposed in [18] and [25]. In [18] the authors propose a general encoding framework for deriving inter-case features. This framework relies on the general notion of *case-types* which can be used to partition an event log into groups of cases that share a common characteristic (i.e. case type). Then, a derivation function is used to obtain representations of inter-case dependencies. The authors evaluate this framework by considering different definitions of case-types and compare the performance of models which are allowed to use inter-case features to models that solely rely on intra-case features in order to make predictions. The best performing model is obtained when case-types are defined based on a prefix consisting of the last three observed events of active cases and the derivation function set to count the number of active cases of each case-type. In [19], the authors extend on the work presented in [18] by considering a data-driven approach for obtaining a representation of inter-case dependencies where proximity measures are used to identify relevant concurrently running cases when making predictions for a case of interest. Similarly, [25] proposes a general framework for deriving inter-case features. This framework relies on considering a number of process perspectives, i.e. control-flow, resource, time and attribute, compared to the very general notion of case-types. The authors then derive inter-case features for each perspective, e.g. the number of cases associated with a specific resource in a user specified time window.

In this paper we extend on the current body of PPM research by proposing a system load based framework for developing inter-case features. To their benefit, previously proposed frameworks for constructing inter-case features are general and multiple different features can be considered and constructed within the proposed frameworks. The framework proposed in [18] for example relies on the very general notion of case types and derivation functions in order to derive inter-case features. Similarly, the framework proposed in [25] relies on a number of process perspectives and multiple inter-case features can be considered and constructed for each of these perspectives. However, the very general nature of these frameworks makes them not straightforward to implement. In comparison, the framework for encoding a multi-location load state proposed here is straightforward to implement for event logs in general. The proposed framework relies on computing system loads at relevant locations in the business process under investigation in order to enrich events with a representation of the multi-location load state of a business process. Two alternative configurations are considered in order to derive the status at relevant locations in a business process. Firstly, a global system based configuration is considered where each event is enriched with information on the status at multiple important locations in the system. Secondly, a case specific configuration is considered where events are enriched with information on the status of the system in close proximity to the case of interest. In order to derive the status of the system in a specific location two derivation functions are considered and compared. The first computes the number of active cases and has been previously considered for deriving inter-case features (see e.g. [8, 18]). The second relies on computing the number of cases which have recently visited the location for which we want to derive the status of. A similar function was considered in [25] where the authors utilize a user specified time window in order to derive inter-case features. However, we extend on this approach by proposing a method for automatically identifying an optimal time window (duration) for the different locations considered.

## 3. Preliminaries

PPM techniques are developed using previously recorded events representing executions of activities in a business process [8]. An event is a tuple $e = (c, t, a, d_1, ..., d_m)$ with $c$ a case identifier, $t$ a timestamp, $a$ an activity label and $d_1, ..., d_m$ a number of additional attributes with $m \geq 0$. A labeling function maps an event to the value of one of its attributes. Let $Case : e \rightarrow c$, $Activity : e \rightarrow a$, $Time : e \rightarrow t$ and $Attribute_m : e \rightarrow d_m$ be labeling functions that return the case identifier, activity label, timestamp and the attribute(s) for a given event. Given these attributes, events can be grouped into traces which are ordered sequences of events for each case where each event refers to a task. More formally, a trace can be defined as $\sigma = \langle e_1, ..., e_n \rangle$, where $n$ is the length of the trace and $e_i \in \sigma$ is the event at position $i$ in trace $\sigma$. It holds that $\forall e_i, e_j \in \sigma, i < j : Time(e_i) < Time(e_j) \wedge Case(e_i) = Case(e_j)$, i.e. the events in the trace are ordered w.r.t. time and all events in the trace share the same case identifier. An event log can then be defined as a set of traces $L = \{\sigma^1, ..., \sigma^K\}$ where $K$ is the total number of traces and $\sigma^m \in L$ is the $m^{\text{th}}$ trace in the event log $L$.

In order to make prediction about the future of an ongoing case, PPM techniques extract information from the execution of historical cases. Since, predictions should be made at any point in a case's development these techniques rely on partitioning traces into a set of prefixes. Given a trace $\sigma = \langle e_1, ..., e_n \rangle$ and two positive integer $k < n$ and $w < n$, we define $Prefix(\sigma, k, w) = \langle e_{k-w+1}, ..., e_k \rangle$ which returns a prefix consisting of $w$ number of events. Typically, PPM techniques rely on labeling functions to derive a relevant feature encoding from obtained prefixes, i.e. in order to extract the activity labels and other relevant attributes of events. Additionally, other functions are often defined to engineer features from the raw event attributes, e.g. a function that extracts the hour of the day from the timestamp of an event or the time which has elapsed since the last activity was executed for that case.

## 4. A framework for encoding the multi-location load state of a business process

By focusing on intra-case features, a large majority of PPM techniques ignore inter-case dynamics when making predictions for ongoing cases. Consequently, these approaches unrealistically assume that cases are processed in isolation from other cases and that the processing of cases is independent of the status of the business process where they are being processed. This limitation has been recognized in [18] and [25] where two general frameworks for constructing inter-case features are proposed. Due to their general nature, almost any type of inter-case feature can be considered and constructed within these frameworks. The framework proposed here solely relies on enriching events with a multi-location load state of relevant locations in a

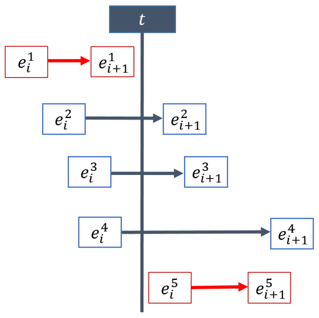

**Figure 1:** A graphical representation of the approach used to count active number of cases in a considered location for time point $t$ derived for an event of interest (i.e. $t = Time(e_t^m)$). In the figure three cases (in blue) are considered active in a location ($Location$) at a given time-point ($t$), e.g. $e_i^3(Time(e_i^3) < t, Activity(e_i^3) = Location)$ and $e_{i+1}^3(Time(e_{i+1}^3) > t, Activity(e_{i+1}^3) \neq Location)$. Additionally, two examples (in red) are given of cases which are not considered active in location at the given time-point. One where the processing of an event occurs before the given time-point, i.e. $Time(e_{i+1}^1) < t$, and another where the processing of an event occurs after the given time-point, i.e. $Time(e_i^5) > t$.

business process where locations are defined based on control-flow information obtained from the process under investigation.

## 4.1. Deriving the load state for a single location in a business process

Two approaches are considered and compared in order to obtain a representation of the load state at a relevant location in a process. Both approaches rely on computing the loads at relevant locations of processes. However, they differ with respect to specifying the duration considered when computing the load for a given location.

**Number of active cases in a location.** The first approach considered for deriving the status at a specific location provides a snapshot view of the number of cases currently being processed at the location under investigation. It therefore, relies on counting the number of active cases at a given location for a given time-point. Hereto, we simply count all currently active cases which most recent event's activity label matches the location for which a load will be computed (i.e. $Activity(e_i^m) = Location$) and were executed before this specific time-point (i.e. $Time(e_i^m) < t$). This is visualized in Figure 1. In the figure, three cases (given in blue) match the conditions outlined above, i.e. their most recently observed activity label matches the load location of interest and are currently being processed at that location.

**Number of cases in an optimized time window.** The second approach considered for deriving the status at a specific location counts the number of cases which have recently passed through a specific location of interest. In order to count the number of cases which have recently been processed at a location a duration needs to be specified for each location. To specify this duration we first compute both the median throughput time and the 95th percentile throughput time value for the location of interest. Then a search is carried out for an optimal duration value in a range defined by the median and 95th percentile value for each location. To find the optimal

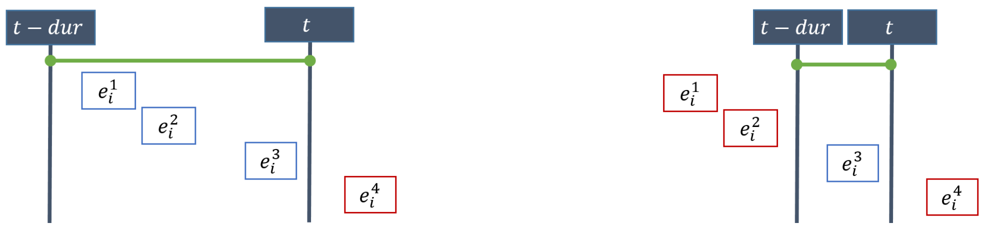

**Figure 2:** A graphical representation illustrating the approach used to count number of recently observed cases in a location of interest. Given an event (i.e. $e_m^i$) an optimal duration (given in green) for location of interest is identified (i.e. $dur = Duration(Location, Activity(e_m^i))$). The endpoints of the considered time window are then defined by the timestamp of the event of interest (i.e. $t = Time(e_m^i)$) and the starting point is defined by the considered duration, (i.e. $t - dur$).

duration for each load location we utilize a random forest trained to predict the remaining runtime of events based on the loads obtained for a specific location using each candidate duration. This optimization is carried out separately for all activity labels of events in order to identify the optimal duration for each combination of load location and activity label. Then, in order to compute the load at a specific location for a given event we simply extract the time stamp of that event and its activity label and identify the optimal duration for that combination of load location and activity label (i.e. $Duration(Location, Activity(e_i^m))$. In order to compute the load for an event of interest we simply count all events which event's activity label matches the location for which a load will be computed (i.e. $Activity(e_i^m) = Location$) and fall in a time window which endpoints are determined by the time stamp of the event and the optimal duration for that location and activity label (i.e. $[Time(e_i^m) - Duration(Location, Activity(e_i^m)), Time(e_i^m)]$).

## 4.2. System based multi-location load state

As mentioned here above cases are not processed in isolation and their processing can be influenced by the status at important locations in a business process. As an example one can think of a recently created loan application which needs to be evaluated. The time it will take to process this application might be affected by the current or recent number of other loan applications that have been evaluated in the this loan application process. Additionally, the case might be affected by the number of cases at a later process step, e.g. the number of loan applications that have already been approved but for which an loan offer has yet to be created. Therefore, the processing of this loan application is not solely affected by intra-case attributes (e.g. previous activities carried out for that case) but also the current or recent status at important locations in the process. The system based inter-case encoding enriches events with information on the status at important locations in a process under investigation. In order to identify important locations we first compute the throughput time of different locations in the process and discard automatic locations. Additionally, infrequent locations (i.e. activities which are carried out for $< 1\%$ of the total number of events) are discarded. All events are then enriched with a multi-location load state containing the load state at important locations in the system.

### 4.3. Case based multi-location load state

The system based inter-case encoding discussed here above provides a representation of the general status at important locations in the system. However, a case might mainly be affected by the load state at locations in the process which are in close proximity to the current location of the case. In order to obtain a representation of the status of the process in close proximity to a case of interest a multi-location load state is derived for locations which are connected to the current location of a case. Firstly, the processing of a case at a given location might be affected by the status at other locations that the case has previously visited. Therefore, we consider the load state at the last location where a case was processed. Secondly, the processing of a case might be affected by the load state at the location where the case is currently being processed. Lastly, the processing of a case of interest might be affected by the status at the locations which the case is most likely to visit next given its current location. In order to derive the load state at likely next locations we therefore compute the state for the five most likely next location given the current location of a cases where e.g. the most likely next location is the location which is most frequently visited next by cases which are processed at the current location of that case. All events are therefore enriched with a multi-location load state containing the load state of the last location where the case was processed, the load state of the current location of that case and the load state of the five location where the case is most likely to be processed next.

## 5. Experimental evaluation

### 5.1. Event logs

A number of event logs were considered when evaluating the multi-perspective framework proposed here. An overview of the considered event logs is given in Table 1. Two of event logs have been made available in relation to the BPI Challange (BPIC), i.e. the BPIC12[1] and BPIC17[2]. The BPIC12-Sub event log is a filtered version of the BPIC12 event log where repetitive events (i.e. sharing the same activity label as the previously observed event) are removed. The BAG event log was was retrieved from the the luggage system of a Brussels Airport. Information on the characteristics of the different event logs is provided in the table. As can be seen the considered event logs are quite varied in terms of the number of available cases (Cases), average trace length (Avg. TL), processing times (Avg. Duration) and number of possible activities executed (Numb. Locations). We also report on the number of important locations (Numb. Imp. Locations), i.e. where non-automatic frequent activities are executed. The system based inter-case encoding scheme discussed here above enriches events with the multi-location load state at these important locations. Time independent data splits between training and test set were utilized when evaluating the performance of the considered models. Hereto, cases in the event logs were order based on their timestamps. Then, the last occurring quarter of cases was considered a test set. The remaining cases which had been processed when the first case in the test set started its process execution were then chosen as a training set.

[1] 10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f
[2] 10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b

**Table 1**
Considered event logs

| Event log | Sector | Cases | Avg. TL | Avg. Duration | Numb. Locations | Numb. Imp. Locations |
|---|---|---|---|---|---|---|
| BPIC17 | Finance | 31509 | 18 | 31646 (min) | 28 | 14 |
| BPIC12 | Finance | 13087 | 20 | 12037 (min) | 26 | 5 |
| BPIC12-Sub | Finance | 13087 | 12 | 12037 (min) | 26 | 5 |
| BAG | Operations | 432357 | 5 | 1187 (sec) | 60 | 9 |

## 5.2. Implementation

The proposed framework for encoding a multi-location load state was evaluated using a conventional PPM setup. Here, multiple prefixes were extracted from all traces in each event log using a window length of 3 and left padded where necessary (i.e. $Prefix(\sigma, k, 3)$ for $k$ in $\langle 1, ..., n\rangle$). Then, a feature vector was extracted from the obtained prefix and propagated to a random forest regressor [26] trained to predict the remaining processing time of a case given a feature vector. Four versions of feature vectors were considered.

- *Intra-case vector* solely contains intra-case information. In order to construct this feature vector, activity labels for all events in the obtained prefixes were extracted and one-hot encoded. Time related information (i.e. the day of the week and hour of the day) was additionally extract and included in this feature vector.
- *System based multi-location load state vector* contains a multi-location load state at important locations in the system obtained using the system based inter-case encoding scheme described here above in addition to the considered intra-case features.
- *Case based multi-location load state vector* was constructed using the case based inter-case feature encoding described here above. It therefore contains the multi-location load state of locations in close proximity to a case of interest in addition to the considered intra-case features.
- *System and case based multi-location load state vector* was additionally considered which includes all intra and inter-case features described here above.

The two approaches for deriving the load state at a location, i.e. counting the active number of cases and counting the number of cases in a optimal time window, were both considered and compared for all models which utilize inter-case features when predicting the remaining processing time of cases.

## 5.3. Results

The results are summarized in Table 2. Our proposed encodings are compared to the baseline scenario of only relying on intra-case features. The mean absolute error (MAE) was used to measure the performance of models. We report on the prediction error in minutes for all event logs with one exception, namely the BAG event log, where we report on the error in seconds since the cases in that event log in general take considerably shorter time to process (as can be seen from Table 1).

**Table 2**
Prediction results for remaining time prediction evaluated using MAE. Models which outperform the baseline intra-case feature model are given in bold. Additionally, an underscore is used to indicate the overall best performing model for each event log.

| Feature vector | BPIC17 | BPIC12-Sub | BPIC12 | BAG |
|---|---|---|---|---|
| Intra-case vector | 10567 | 7988 | 8591 | 707 |
| **System based multi-location load state vector** | | | | |
| Number of active cases | 10788 | 8562 | **8573** | **704** |
| Number of cases in optimized time window | **10548** | **7910** | **8387** | **703** |
| **Case based multi-location load state vector** | | | | |
| Number of active cases | **10527** | **7849** | 9041 | **691** |
| Number of cases in optimized time window | **10520** | **7947** | **8396** | **690** |
| **System and case based multi-location load state vector** | | | | |
| Number of active cases | 10822 | 8076 | 9076 | **704** |
| Number of cases in optimized time window | **10520** | **7885** | **8361** | **699** |

A general performance gain can be observed for models that use inter-case features compared to models that solely rely on using intra-case features when predicting the remaining processing time of cases. As can be seen from the table, models that utilize a feature vector which contains some version of the multi-location load state encoding proposed here is the best performing model on all event logs considered. More specifically, a model that uses a case based multi-location load state vector, containing information on the load states of locations connected to the current location of a case of interest in addition to intra-case features, is the best performing model on all considered event logs, with one exception being the BPIC12 event log where a model that uses both system and case based multi-location load state vector in order to make predictions performs best.

When the two approaches for deriving the load state at a given location are compared, one can see that counting the number of recently observed cases in a time window which is optimized for all combinations of locations for which a load is computed and activity labels of events in general leads to a better performance compared to counting the number of active cases in a load location. This is the case for all event logs and versions of the multi-location load state vectors considered here with one exception, namely a model trained using a case based multi-location load state vector on the BPIC12-Sub event log. Here, considering the number of active cases leads to better performance for predicting the remaining processing time of cases compared to considering the number of cases in a optimized time window.

## 6. Discussion

A general performance increase was observed for models that are allowed to utilize inter-case features, developed using the proposed framework for encoding a multi-location load state, when predicting the remaining processing times of cases compared to models that solely rely on using intra-case features for this prediction task. From Table 1 it can be observed that the average processing time for cases in the BAG event log is considerably shorter then for other event logs. This event log was obtained from the luggage system at a large international airport. The system is almost fully automated, with bags being scanned at each location when moving through the system. From the results, we can observe a relatively large performance gain for this event log when models are allowed to utilize inter-case features obtained using the framework proposed here. Additionally, it can be observed that all considered versions of multi-location load state vectors and approaches for deriving the load state at a location lead to an improved performance compared to solely relying on intra-case features when predicting the remaining processing time for cases in this event log. To some extent this is to be expected given the nature of this process, i.e. it seems reasonable that the processing of bags in a luggage handling system is largely affected by the load state at relevant parts of the system in addition to intra-case features. Other event logs considered were extracted from loan application processes of banks. As can be seen the best performing model for all of these event logs was obtained using a version of a multi-location load state vector. Therefore, incorporating load state at relevant locations in the considered loan application processes has a beneficial impact. However, the processing time of loan applications is also largely influenced by external factors which cannot be represented by typical intra-case features nor inter-case features. The processing of a loan application will for example to some extent depend on the willingness of the loan applicant to participate in the process, e.g. by correctly filling out his loan application and responding to a loan offer. Therefore, the processing does not solely depend on typical intra-case features or inter-case features but also other contextual information which is not incorporated by the multi-location load state encoding framework proposed here.

## 7. Conclusion

The framework proposed in this paper encodes a multi-location load state representation at relevant locations in a business process. This framework solely relies on deriving the load state at relevant locations and therefore provides users with a straightforward approach for incorporating the status of a business process in order to make improved predictions when developing PPM techniques. This framework was evaluated using a conventional PPM setup where random forest regression models were trained to predict the remaining processing times of cases. The models were constructed using three different feature vectors obtained using the proposed framework and their performance compared to a model which solely relies on intra-case features in order to make prediction. This comparison was carried out over four real-life event logs. A general performance gain was observed for models that use inter-case features encoded using the multi-location load state framework proposed here. More specifically, models that used a case based multi-location load state vector in addition to intra-case features performed best on all considered event logs with one exception where a model that uses both system and case based multi-location load state performed best.

The proposed framework relies on encoding the current or recent load state at relevant locations in a process under investigation. However, the future status of a process might also have an impact on the processing of cases. It would therefore be of interest to expand on the framework proposed here by incorporating predictions for the future multi-location load state of a process under investigation. Additionally, LSTMs [27] have been shown to perform well for a number of PPM tasks including predicting the remaining processing time of cases. It would therefore be of interest to investigate the additional value of incorporating inter-case features encoded using the multi-location load state framework proposed here in a PPM setup which uses an LSTM for this prediction task. Lastly, this paper evaluated the additional value of inter-case features obtained using the proposed framework for predicting the remaining processing time of cases. It would be of interest to expand on this by evaluating the additional value of incorporating information encoded using the proposed framework for other prediction tasks.

## References

[1] I. Teinemaa, M. Dumas, A. Leontjeva, F. M. Maggi, Temporal stability in predictive process monitoring, Data Mining and Knowledge Discovery 32 (2018) 1306–1338.

[2] F. M. Maggi, C. D. Francescomarino, M. Dumas, C. Ghidini, Predictive monitoring of business processes, in: International conference on advanced information systems engineering, Springer, 2014, pp. 457–472.

[3] M. Dumas, M. La Rosa, J. Mendling, H. A. Reijers, et al., Fundamentals of business process management, volume 1, Springer, 2013.

[4] C. Di Francescomarino, C. Ghidini, F. M. Maggi, F. Milani, Predictive process monitoring methods: Which one suits me best?, in: International conference on business process management, Springer, 2018, pp. 462–479.

[5] W. M. Van der Aalst, M. H. Schonenberg, M. Song, Time prediction based on process mining, Information systems 36 (2011) 450–475.

[6] A. Rogge-Solti, M. Weske, Prediction of business process durations using non-markovian stochastic petri nets, Information Systems 54 (2015) 1–14.

[7] I. Verenich, M. Dumas, M. L. Rosa, F. M. Maggi, I. Teinemaa, Survey and cross-benchmark comparison of remaining time prediction methods in business process monitoring, ACM Transactions on Intelligent Systems and Technology (TIST) 10 (2019) 1–34.

[8] I. Teinemaa, M. Dumas, M. L. Rosa, F. M. Maggi, Outcome-oriented predictive process monitoring: Review and benchmark, ACM Transactions on Knowledge Discovery from Data (TKDD) 13 (2019) 1–57.

[9] W. Kratsch, J. Manderscheid, M. Röglinger, J. Seyfried, Machine learning in business process monitoring: a comparison of deep learning and classical approaches used for outcome prediction, Business & Information Systems Engineering 63 (2021) 261–276.

[10] J. Wang, D. Yu, C. Liu, X. Sun, Outcome-oriented predictive process monitoring with attention-based bidirectional lstm neural networks, in: 2019 IEEE International Conference on Web Services (ICWS), IEEE, 2019, pp. 360–367.

[11] M. Ceci, P. F. Lanotte, F. Fumarola, D. P. Cavallo, D. Malerba, Completion time and next activity prediction of processes using sequential pattern mining, in: International Conference on Discovery Science, Springer, 2014, pp. 49–61.

[12] N. Tax, I. Verenich, M. L. Rosa, M. Dumas, Predictive business process monitoring with lstm neural networks, in: International Conference on Advanced Information Systems Engineering, Springer, 2017, pp. 477–492.

[13] J. Evermann, J.-R. Rehse, P. Fettke, Predicting process behaviour using deep learning, Decision Support Systems 100 (2017) 129–140.
[14] C. Di Francescomarino, C. Ghidini, F. M. Maggi, G. Petrucci, A. Yeshchenko, An eye into the future: leveraging a-priori knowledge in predictive business process monitoring, in: International conference on business process management, Springer, 2017, pp. 252–268.
[15] M. Polato, A. Sperduti, A. Burattin, M. d. Leoni, Time and activity sequence prediction of business process instances, Computing 100 (2018) 1005–1031.
[16] M. Camargo, M. Dumas, O. González-Rojas, Learning accurate lstm models of business processes, in: International Conference on Business Process Management, Springer, 2019, pp. 286–302.
[17] V. Denisov, D. Fahland, W. M. van der Aalst, Unbiased, fine-grained description of processes performance from event data, in: International Conference on Business Process Management, Springer, 2018, pp. 139–157.
[18] A. Senderovich, C. Di Francescomarino, C. Ghidini, K. Jorbina, F. M. Maggi, Intra and inter-case features in predictive process monitoring: A tale of two dimensions, in: International Conference on Business Process Management, Springer, 2017, pp. 306–323.
[19] A. Senderovich, C. Di Francescomarino, F. M. Maggi, From knowledge-driven to data-driven inter-case feature encoding in predictive process monitoring, Information Systems 84 (2019) 255–264.
[20] V. Denisov, D. Fahland, W. M. van der Aalst, Predictive performance monitoring of material handling systems using the performance spectrum, in: 2019 International Conference on Process Mining (ICPM), IEEE, 2019, pp. 137–144.
[21] R. Conforti, M. de Leoni, M. La Rosa, W. M. van der Aalst, A. H. ter Hofstede, A recommendation system for predicting risks across multiple business process instances, Decision Support Systems 69 (2015) 1–19.
[22] F. Folino, M. Guarascio, L. Pontieri, Discovering context-aware models for predicting business process performances, in: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems", Springer, 2012, pp. 287–304.
[23] E. L. Klijn, D. Fahland, Identifying and reducing errors in remaining time prediction due to inter-case dynamics, in: 2020 2nd International Conference on Process Mining (ICPM), IEEE, 2020, pp. 25–32.
[24] M. Pourbafrani, S. Kar, S. Kaiser, W. M. van der Aalst, Remaining time prediction for processes with inter-case dynamics, in: International Conference on Process Mining, Springer, Cham, 2021, pp. 140–153.
[25] A. Grinvald, P. Soffer, O. Mokryn, Inter-case properties and process variant considerations in time prediction: A conceptual framework, in: Enterprise, Business-Process and Information Systems Modeling, Springer, 2021, pp. 96–111.
[26] L. Breiman, Random forests, Machine learning 45 (2001) 5–32.
[27] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (1997) 1735–1780.