IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 26, 2019, accepted August 7, 2019, date of publication September 5, 2019, date of current version September 20, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2939631

A Vector-Based Classification Approach for Remaining Time Prediction in Business Processes

AHMAD ABUROMMAN, MANUEL LAMA, AND ALBERTO BUGARÍN

Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS), Universidade de Santiago de Compostela, E15782 Santiago de Compostela, Spain

Corresponding author: Alberto Bugarín (alberto.bugarin.diz@usc.es)

This work was supported in part by the Spanish Ministry for Science, Innovation, and Universities under Grant TIN2017-84796-C2-1-R, and in part by the Galician Ministry of Education, University, and Professional Training under Grant ED431C 2018/29 and "accreditation 2016-2019, Grant ED431G/08." All grants were co-funded by the European Regional Development Fund (ERDF/FEDER program).

ABSTRACT In this paper, we deal with one of the current challenges in process mining enhancement: the prediction of remaining times in business processes. Accurate predictions of the remaining time, defined as the required time for an instance process to finish, are critical in many systems for organizations being able to establish *a priori* requirements, for optimal management of resources or for improving the quality of the services organizations provide. Our approach consists of *i)* extracting and assessing a number of features on the business logs, that provide a structural characterization of the traces; *ii)* extending the well-known annotated transition system (ATS) model to include these features; *iii)* proposing a partitioning strategy for the lists of features associated to each state in the extended ATS; and *iv)* applying a linear regression technique to each partition for predicting the remaining time of new traces. Extensive experimentation using eight attributes and ten real-life datasets show that the proposed approach outperforms in terms of mean absolute error and accuracy all the other approaches in state of the art, which includes ATS-based, non-ATS based as well as Deep Learning-based approaches.

INDEX TERMS Business processes enhancement, predictive business process monitoring, business processes management, business intelligence.

I. INTRODUCTION

Massive growth of business processes automation as well as increasing information technology adoption in business process management is producing a vast amount of process execution data that are stored in the form of event logs [1], [2]. By applying process mining techniques to these logs [3], real hidden processes can be discovered [4], [5] and/or existing processes can be monitored and improved [6].

There are three main types of process mining techniques [7]: process discovery, conformance checking, and process enhancement. Process discovery takes an event log and produces a model without using *a priori* information [3], [7]. Conformance checking makes a comparison between a designed process model and the process discovered from the event log, to show where the real process deviates from the designed one [7]. Process enhancement aims to extend or improve an existing process, using information related to the process which is usually extracted from the recorded events logs [8].

In process enhancement, temporal information is usually used to measure the waiting times among process activities, to check the temporal behaviour during traces replay, to provide information about relevant issues in the process (e.g., bottlenecks, throughout times, frequencies) or to predict the remaining times from running process instances [7]. In this sense, predicting the remaining time of process instances (running cases) has been highlighted in the literature as one of the most important current challenges in process mining [9]. Remaining time of a process instance is the required time for it to be finished from a particular execution state. Accurately predicting remaining time is a key issue for all actors around business processes management. For organisations, having accurate time predictions allows them to manage the resources efficiently [10], to assess the quality of the services they provide as well as to take

The associate editor coordinating the review of this manuscript and approving it for publication was Khursheed Aurangzeb.

appropriate managerial decisions in advance. For end-users, it is also critical to be aware about when the processes they are involved in will finish [1], [4]. Some examples of the latter are bank customers applying for a loan, who need to know in advance how long will it take for their loan application to be reviewed, checked, assessed and accepted or declined, or health treatment processes, where it is crucial to know the remaining time of each treatment to effectively manage the next treatments (for preparing in advance all the necessary resources) or the next patients.

The problem of predicting the remaining time is a part of a more general problem known as predictive monitoring. In the last years, several proposals focused on predictive monitoring and, more specifically, on the prediction of remaining time have been presented [11]–[13]. Initially, these proposals have focused on the representation of the process executions or traces under the hypothesis that traces with different characteristics have different remaining times. Several of these approaches are based on Annotated Transition Systems (ATS), where each (partial) trace is associated to a state [4]–[6], [14], [15]. Other approaches use a partial trace-based or index-based representation [1], [10], [16]. More recently, approaches have been proposed for applying machine learning methods for predicting the remaining time [2], [17]–[20]. In all these approaches, the problem encoding includes information about the context of the process execution state, such as the duration of the activities, or about domain variables. The main problem with all these approaches is that their trace representation (or encoding) does not include all the relevant information related to the traces execution, such as repetition of activities, the distance between activities or co-occurrences. Without this information about the structural features of the traces, it is difficult to make accurate predictions about the remaining time.

In this paper, we present a new vector-based extended ATS-based approach that considers structural features or attributes related to the process execution. In our approach, each state in the ATS is annotated with a list of vectors which contain information related to, for instance, frequency of activities frequency, repetition of activities (loops), activities distance, and others. The lists of vectors are partitioned according to a similarity criteria of the remaining times of the traces in it. For each partition, a linear regression-based predictor is built, thus achieving that the prediction takes structural information of the traces into account. Our approach has been compared to other approaches and validated with ten real-life datasets, obtaining more precise predictions than all the proposals of the state of the art [13].

The paper is structured as follows: in Section II the main approaches in state of the art for remaining time prediction are discussed; in Section III the main concepts of our approach are presented; in Section IV our prediction model for remaining time estimation is described; in Section V the experiments for validating our prediction model are described and the main results are discussed, and, finally, Section VI summarises the main contributions of the paper.

II. RELATED WORK

As introduced in Section I, the problem of predicting the remaining time of business processes has been addressed in several proposals, which have been discussed in very recent surveys [11]–[13]. The models described in [5], [6], [14], [15] are state-based representations entitled Annotated Transition Systems (ATS). In Transition Systems, the process traces (real executions or instances) are represented as a sequence of states and a set of transitions between them: a state models a sequence of activities of the trace and a transition represents the execution of the next activity in the trace. Each state of this Transition System is annotated with temporal information about the process execution, thus generating an Annotated Transition System (ATS). More specifically, in [6], [14] authors annotate each state with the average of the remaining time to complete each trace execution represented by this state. In this model, this average time is provided as the remaining time prediction. In [5] authors build an ATS enriched with both classification and regression models: for each state, a Naive Bayes classifier is built for predicting the probability of transition from this state to the following one, and a support vector regression approach is used for predicting the remaining time for this second state. Both machine learning models are trained with values of data attributes generated during the case execution. These attributes contain temporal references and domain-specific information. In [4] authors use a Hidden Markov Model (HMM) to obtain the transition probability from one state to the following one. Each state in the ATS is annotated with a weighted average of the probabilities given by the HMM where the weights for each state are the average of the remaining times values of the traces that fit this state. In [15] an ATS containing only frequent parts of traces is built. These frequent parts are extracted by means of a sequential pattern mining algorithm. Therefore, the ATS states do not model all the sequences of traces, but the frequent parts of them. Each state of this ATS is annotated with the resource performing each activity, the cost associated with the activity and the prefix where the activity appears. Then, for each state, a regression model based on these attributes is used to predict the remaining time.

Other works do not follow an ATS-based approach. In [10] authors present an approach based on query catalogues, which are groups of partial trace tails from all the traces available in a log. A partial trace tail is annotated with the number of its occurrences and the sum of its remaining times. Then, the prediction of the remaining time to completion for each partial trace is the average time in the catalogue where the trace is in. In [1] authors present a prediction method based on non-Markovian Petri Nets, which are enriched with duration distributions of activities and the elapsed time since the occurrence of the last activity. In [16] authors propose a white-box approach to predict the remaining time of running process instances. The approach followed is firstly to predict the remaining time at the level of activities and then to aggregate these predictions at the level of a process instance by means of flow analysis techniques. To encode the process

traces an index-based encoding is considered, where for each position in a trace the event occurring at that position and the value of each event attribute are considered.

More recent approaches use machine learning methods to predict the remaining time [11], [12], [16]. These models, in general, produce better results than the previously described approaches. In [2] authors built four estimators based on regression techniques, each of which makes use of different parameters: average duration of the traces, duration of the activities, the occurrence of the activities, and domain attributes related to parameters or variables obtained during the execution cases. The best estimator is the attribute-based estimator, which suggests that the remaining time prediction depends on the values of the variables generated during the process execution. In [17] authors present an approach in two phases: first, prefixes of previous cases are clustered according to control flow information, mainly related to the activity frequency; and then a classifier is built for each cluster using event data attributes to discriminate between cases that lead to a fulfilment of the predicate under examination and cases that lead to a violation within the cluster. At runtime, a prediction is made on a running case by mapping it to a cluster and applying the corresponding classifier. In [18] authors propose to use LSTM neural networks to predict the next event of a running case and its time-stamp. The encoding of the neural network includes the type of activity and the time between two consecutive events. Based on this neural network, the remaining time is estimated by iteratively predicting the next activity and its duration until a special end of case activity is predicted. In [19] authors present a context-aware clustering-based approach to the discovery of predictive models for supporting the forecast of process performance measures, such as the remaining time, where major performance-relevant variants of the process are related to different regression models and discriminated through context variables. In this approach, each trace is represented through a vector that includes the activity frequency and some properties related to the process execution context. In [20] authors present a method that represents process instances by considering both intra-instance dependencies and inter-instance dependencies, where shared information between process instances is taken into account. Based on this encoding two different machine learning methods, linear regression, and random forest, are applied to predict the remaining time.

The main problem with these techniques is that the coding they use does not sufficiently represent or characterise all the information about the traces. Most of these techniques only take into account the frequency of occurrence of activities and/or activities that occur in the context of an activity, i.e., activities that appear in a window of occurrence around an activity. Some techniques complete the coding of the traces with domain attributes related to variables obtained during the execution of the traces, with the duration of the activities or with the time remaining to complete the trace execution. However, in these approaches trace encoding does not explicitly represent the complex relationships between traces; in particular, the repetition of the same activity (1-size loop), the repetition of several activities (n-size loops), the co-occurrence of any two activities (although this characteristic is considered in approaches based on deep learning) or the distance between two activities (i.e., distance between their indexes in a trace). Thus, in these approaches, for instance, the encoding of a trace in which activity appears five-times with different indexes is the same as the encoding of a trace where the activity appears five-times in a row (1-size loop). This same encoding for traces that are structurally different prevents current approaches to make very accurate predictions of the remaining time. In these cases, which are very common in real problems, considering structural information about the traces, is a need which is considered our model. Experimental results described in Section V show the appropriateness of our approach.

III. PRELIMINARIES

This section describes the required elements to build our prediction model, which consists of two parts; firstly, to build an extended annotated transition system which includes a number of attributes that comprise relevant structural information about the traces. Secondly, to apply regression techniques for predicting the remaining time of the process execution at each time.

Definition 1 (Event) [6]: An event e is described by a unique identifier and is characterised by its properties, such as its identifier, time-stamp, and the activity which is executed in the time-stamp.

Definition 2 (Trace, Event Log) [6]: A trace T is a sequence of ordered events $\{e_1, e_2, \ldots, e_N\}$. An event log is a set of traces.

Definition 3 (Partial trace, State) [6]: A partial trace PT is any continuous part of a trace T that contains one or more events in sequence. For each PT, three state representations are defined [6]: Set, Multi-Set, and Sequence. In this paper, we focus on the Set representation as the basis for our model. In Set representation, each partial trace PT has associated a state $S(PT)$ which is labelled through the activities in PT and where no repetition of activities is considered (no matter its execution order).

Definition 4 (Transition System, TS) [6]: A transition system TS is a triplet (S, PT, TR), where S is the state space, PT is a set of partial traces, and TR is a transition relation which describes how the system moves from one state to another state. A TS has different forms depending on the state representation in which it is based on.

In Fig. 1 we show the differences between Sequence representation and Set representation in a TS, using a simple example of a process involving only three events with its corresponding activities (A, B, C). We consider the representation of two different traces ($\langle$ABC$\rangle$ $\langle$ACB$\rangle$) involving the three activities A, B and C in different order. In Fig. 1a),

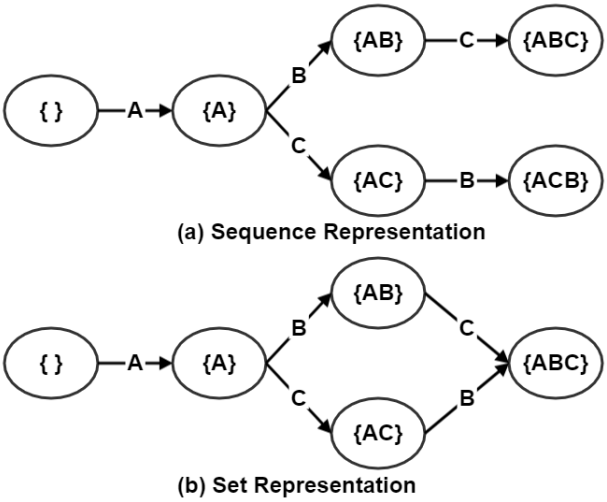

FIGURE 1. Transition system with (a) sequence representation, and (b) set representation of traces ⟨ABC⟩, and ⟨ACB⟩ (see Section III).

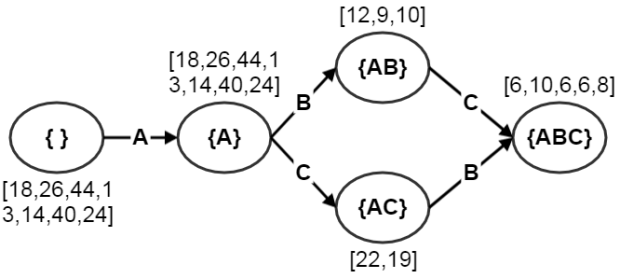

FIGURE 2. ATS-based on the log shown in Table 1 and Section III.

we show the Sequence representation of both traces, where the final state is different for each trace, respectively {ABC} and {ACB}. In Fig. 1b, the final state for both traces is the same (state {ABC}), since Set representation does not consider the order of the activities within the trace. For example, other traces involving the same activities, like ⟨CBA⟩, or ⟨BAC⟩ are represented by the same state {ABC}.

Definition 5 (Annotated Transition System, ATS) [6]: An Annotated Transition System (*ATS*) is a duple (*TS*, *MF*), where *TS* is a transition system based on the Set representation and *MF* is a measurement function *MF*(*S*) that annotates each state *S* in the *ATS*. For instance, in [6], authors define *MF* as the remaining time since the occurrence of the last activity of each partial trace in the *TS* until that trace is completed.

In Fig. 2 we show an example of the ATS model defined in [6], built from the traces described in Table 1. Let us consider state {AB} and the partial traces it represents: $\langle A^{00}B^{06}\rangle$ (from Trace 1), $\langle A^{15}B^{19}\rangle$ (from Trace 4), and $\langle A^{18}B^{22}\rangle$ (from Trace 5). Remaining times are calculated as the time elapsed since the time-stamp of the last activity in each partial trace (6, 19, and 22 respectively) until the end of the corresponding trace (18, 28, and 32, respectively). Therefore, the annotation attached to the state {AB} is [12, 9, 10].

TABLE 1. An example log, showing seven traces, each of them represented as a sequence of activities *A*, *B*, *C*, *D*, *E*, occurring in different orders. Superscript numbers indicate time-stamps at which each activity is completed.

#	Traces
1	$<A^{00}B^{06}C^{12}D^{18}>$
2	$<A^{10}C^{14}B^{26}D^{36}>$
3	$<A^{12}E^{22}D^{56}>$
4	$<A^{15}B^{19}C^{22}D^{28}>$
5	$<A^{18}B^{22}C^{26}D^{32}>$
6	$<A^{19}E^{28}D^{59}>$
7	$<A^{20}C^{25}B^{36}D^{44}>$

IV. A NEW PREDICTION MODEL FOR REMAINING TIME ESTIMATION

In the previous section, we presented the most relevant elements of an ATS. Based on it, we propose and describe in detail in this section our extension of the ATS model that includes structural features which are extracted from the traces.

A. TRACE FEATURES

Firstly, we extend the ATS model by considering a number of features (or attributes) extracted from the analysis of the event log traces. Each of these features is related to a measurement with which the ATS model will be extended, that is, each state of the ATS model will be annotated with both a set of attributes and the remaining time. A key difference between our approach and others in the literature is that the attributes we consider provide specific structural information about traces, such as the occurrence of the activities, its elapsed time, or the existence of loops, among others. This structural information will act as predictor variables that will be taken into account by a regression model, aiming to improve the accuracy in the calculation of the remaining time prediction in a running process.

We define the following eight attributes:

Definition 6 (Occurrence, Occ): Let *PT* be a partial trace. We define *Occ*(A_i,*PT*) as the number of times activity A_i occurs in *PT*. For example, for partial trace *PT* = ⟨CCABBCAA⟩, we have *Occ*(*A*,*PT*) = 3, *Occ*(*B*,*PT*) = 2, and *Occ*(*C*,*PT*) = 3.

Definition 7 (Cycle, Cyc): Let *PT* be a partial trace, and *LargSeq*(A_i) the largest sequence of activity A_i in *PT*. We define *Cyc*(A_i,*PT*) := *length*(*LargSeq*(A_i))-1 as the number of times the activity A_i is repeated in sequence in *PT*. For example, for partial trace *PT* = ⟨CCCABCCAA⟩, we have *Cyc*(*A*, *PT*) = 1, *Cyc*(*B*, *PT*) = 0, and *Cyc*(*C*, *PT*) = 2.

Definition 8 (Position, Pos): Let *PT* be a partial trace *PT*, and *Pos*(A_i) the index set of activity A_i in that partial trace *PT*. We define *Position*(A_i, *PT*) := maximum(*Pos*(A_i)) as the last happening of A_i in *PT*. It represents the last index of an activity happened in the partial trace. For example, for the partial trace *PT* = ⟨CCCABCCAA⟨, we have *Pos*(*A*, *PT*) = 9, *Pos*(*B*,*PT*) = 5, and *Pos*(*C*,*PT*) = 7.

Definition 9 (Distance, Dis): Let A_i be an activity in a partial trace PT. $Dis(A_i, PT)$ (distance) is the number of activities (different of A_i) between the last occurrence of A_i and the previous one (backwards) in PT. $Dis(A_i, PT) = 0$ in case A_i is a single activity in PT. For example, for the partial trace $PT = \langle$CCCABCCA$\rangle$, we have $Dis(A) = 3$, $Dis(B) = 0$, and $Dis(C) = 0$.

Definition 10 (Duple, Dup): Let PT be a partial trace S its associated state in an ATS. For all the pairs of activities A_i and A_j in S, we define $Dup(A_i,A_j,PT)$ as the number of times that the sequence A_iA_j happens in PT. For example, for the partial trace $PT = \langle CACACCAB\rangle$, we have $Dup(A,A,PT) = 0$, $Dup(A,B,PT) = 1$, $Dup(A,C,PT) = 2$, $Dup(B,A,PT) = 0$, $Dup(B,B,PT) = 0$, $Dup(B,C,PT) = 0$, $Dup(C,A,PT) = 3$, $Dup(C,B,PT) = 0$ and $Dup(C,C,PT) = 1$.

Definition 11 (Change): Let PT be a partial trace and LA its last activity. We define $Change(PT)$ as the number of times the activities move from one activity to a different one from the beginning of PT until LA. For example, for the partial trace $PT = \langle CCCABCCA\rangle$, we have $Change(PT) = 4$, since it represents the move from activity C to A (first change), then the move from A to B (second change), then the move from B to C (third change), and finally the move from C to A (fourth change).

Definition 12 (Single): Let PT be a partial trace. We define $Single(PT)$ as the number of single activities that have no cycle from the beginning of PT. For example, for a partial trace $PT = \langle CCCABCCA\rangle$, we have $Single(PT) = 1$, this partial trace has only one single activity (B).

Definition 13 (Elapsed Time, Elt): Let LA be the last activity in a partial trace PT. We define $Elt(PT)$ as the time passed since the beginning of PT until LA. For example, taking the $PT = \langle A^{10}C^{14}B^{26}D^{36}\rangle$ (the second one in Table 1), we have $Elt(PT) = 36$.

All the attributes we consider are related to structural features which are of interest for characterising traces and/or partial traces. For instance, the occurrence of activity is related to the repetitions in traces; cycle and duple are related to the existence of loops in traces; whilst changes of events and their position are related to variety in traces.

In Table 2, we present an illustrative example of the calculation of the previously defined attributes for the example trace ⟨ABBBAABBAB⟩. We can see in this example that some of these attributes (e.g., *Change*, *Single*, *Elt*) produce a single value from each trace. Other attributes (*Occ*, *Cyc*, *Pos*, *Dis*) produce a number of different values which is in linear order with the number of events N in the trace. In the example, $N = 2$, and therefore we have two values for each of the attributes, eight in total. Finally, the last attribute (*Duple*) produces N^2 values (4 in the example, for all the combination pairs of the activities in the trace, A and B). According to this number of attributes values, in case of traces involving a low number of events, no scalability problems occur. However, if the number of events is high, also a high number of attributes will be produced. In order to cope with scalability problem, we will describe in Section V the use of an attribute selection method which reduces the number of attributes by keeping only the attributes that have a real impact on the remaining time estimation and removing the less relevant ones.

TABLE 2. Value of the attributes for the partial trace $PT = \langle A^2B^7B^{13}B^{22}A^{30}A^{37}B^{50}B^{54}A^{60}B^{62}\rangle$, where superscripts of the activities indicate their timestamps.

Attribute	Description	Representation	Att. Value
$Occ(A,PT)$	How many times A occurs.	ABBBAABBAB	4
$Occ(B,PT)$	How many times B occurs.	ABBBAABBAB	6
$Cyc(A,PT)$	The maximum continuous repeat of A	ABBBAABBAB	1
$Cyc(B,PT)$	The maximum continuous repeat of B	ABBBAABBAB	2
$Pos(A,PT)$	The position of last occurrence of A	ABBBAABBAB	9
$Pos(B,PT)$	The position of last occurrence of B	ABBBAABBAB	10
$Dis(A,PT)$	The distance between the last occurrence of A and the previous one	ABBBAABBAB	2
$Dis(B,PT)$	The distance between the last occurrence of B and the previous one	ABBBAABBAB	1
$Dup(A,A,PT)$	Duple of AA	ABBBAABBAB	1
$Dup(A,B,PT)$	Duple of AB	ABBBAABBAB	3
$Dup(B,A,PT)$	Duple of BA	ABBBAABBAB	2
$Dup(B,B,PT)$	Duple of BB	ABBBAABBAB	3
$Change(PT)$	Change of activities	ABBBAABBAB	5
$Single(PT)$	Number of single activities	ABBBAABBAB	0
$Elt(PT)$	Timestamp of the last activity, as annotated in the caption	ABBBAABBAB	62

B. EXTENDED ANNOTATED TRANSITION SYSTEM

Previously, we introduced the definition of an Annotated Transition System (ATS) model, as well as the definitions of the attributes we consider to be extracted from each trace to build our model. In this section, we show how we integrate the previously introduced attributes into an ATS in order to define our Extended Annotated Transition System (EATS).

Definition 14 (List): Let S be a state of an annotated transition system ATS associated to a given set of Partial Traces PT_j, $j = 1, \ldots, P$ (being P the total number of Partial Traces associated to S). Let $\{Att_i, i = 1, 2, \ldots, N\}$ be the attributes defined in Section IV-A, and element $E_j = [valueOfAtt_1, \ldots, valueOfAtt_M, RT_j]$, $j = 1, \ldots, P$ a vector

made up of the attribute values for each PT_j, being M the total number of attribute values and RT_j the remaining time for PT_j (time until trace completion, as defined in Section III). We define $List(S) := \{E_1, E_2, \ldots, E_P\}$ as the set of elements that annotate state S. Note that each state has an associated list which includes P elements (being P, as stated before, the number of partial traces represented by state S).

Considering definitions 6-13, we have that the element E_j associated to a given partial trace PT_j is made up of the following components:

- Occurrence: $Occ(A_i, PT_j)$, ∀ activities A_i in PT_j
- Cycle: $Cyc(A_i, PT_j)$, ∀ activities A_i in PT_j
- Position: $Pos(A_i, PT_j)$, ∀ activities A_i in PT_j
- Distance: $Dis(A_i, PT_j)$, ∀ activities A_i in PT_j
- Duple: $Dup(A_i, A_k, PT_j)$, ∀ pairs of activities A_i, A_k in PT_j
- Change(PT_j)
- Single(PT_j)
- Elapsed time: $Elt(PT_j)$
- Remaining time: $RT(PT_j)$

As an example, let us consider the following partial trace $PT = \langle A^{00}B^{06}\rangle$, which is extracted from the first trace in Table 1. Then, we have that the element E associated to PT is made up of:

- Occurrence: $Occ(A, PT) = 1$, $Occ(B, PT) = 1$
- Cycle: $Cyc(A, PT) = 0$, $Cyc(B, PT) = 0$
- Position: $Pos(A, PT) = 1$, $Pos(B, PT) = 2$
- Distance: $Dis(A, PT) = 0$, $Dis(B) = 0$
- Duple: $Dup(A, A) = 0$, $Dup(A, B) = 1$, $Dup(B, A) = 0$, $Dup(B, B) = 0$
- Change $= 1$
- Single $= 2$
- Elapsed time: $Elt = 6$
- Remaining time: $RT(PT) = 12$

Therefore, element E will be the following vector:

$$[1, 1, 0, 0, 1, 2, 0, 0, 0, 1, 0, 0, 1, 2, 6, 12]$$

Definition 15 (Extended Annotated Transition System, EATS): Let L be an event log, and ATS an annotated transition system obtained based on a Set representation for L. An Extended Annotated Transition System ($EATS$) for L is an extension of ATS which consists in annotating each state S in ATS through a measurement function $MF(S) = List(S)$, where $List(S)$ is the List of elements as defined in Definition 14.

In Algorithm 1, we describe the procedure for formally building our Extended Annotated Transition System. Lines 1 to 3 initialise the Lists of Elements that are associated to each state of the EATS. The loop in (line 4) iterates all partial traces, whilst in line 5 we obtain the associated state to each partial trace. Then, we store the attribute values in the element (lines 7-19), according to Def. 14. The element is completed by adding the remaining time of the partial trace (line 10) and stored in the list associated to the current state (line 11). The procedure is repeated throughout all the partial traces in the EATS and, finally, the Extended Annotated Transition System is returned.

Algorithm 1 Construction of an Extended Annotated Transition System (EATS)

Input: TS = (S, PT, TR): Transition System (Def. 4);
Output *EATS*: Extended Annotated Transition System

1: **for each** $s \in S$ **do**
2: | List(s) $= \varnothing$ ▷ Initialise Lists of Elements
3: **end for**
4: **for each** $pt \in PT$ **do** ▷ For each partial trace
5: | $CS \leftarrow S(pt)$ ▷ State associated to pt (Def. 3)
6: | $E = \varnothing$ ▷ Initialise Element
7: | **for** $i = 1, \ldots, M$ **do** ▷ Def. 14, M # attr. values of pt
8: | | $E \leftarrow E \cup valueOfAtt_i(pt)$ ▷ Add $i-th$ attribute value
9: | **end for**
10: | $E \leftarrow E \cup RT_{pt}$ ▷ Add remaining time of pt
11: | $List(CS) \leftarrow List(CS) \cup E$
12: **end for**
13: $List \leftarrow \{List(s), \forall s \in S\}$
14: $EATS \leftarrow (TS, List)$
15: **return** *EATS*

TABLE 3. Sample log with three traces.

Trace 1	$<A^3B^{10}B^{22}C^{40}A^{45}>$
Trace 2	$<A^2A^{16}C^{19}B^{24}B^{50}>$
Trace 3	$<A^7C^{11}C^{15}C^{24}B^{60}>$

In Fig. 3, we also show an example of how the EATS is built from the traces in Table 3. We associate to each state S a list of elements (vectors), which include the values of the eight attributes defined in Section IV-A calculated for each of the partial traces represented by S. For example, at state {AB}, we have two partial traces Pt12, ⟨AB⟩, and Pt13, ⟨ABB⟩. Each vector in the list includes the corresponding 15 attributes values (plus the value to be predicted, remaining time). Regarding the number of attributes, it is worthy remembering that it differs among the states, since it depends on how many activities engage in that state, as explained before. For instance, in this example, the partial traces associated to one event (State {A}) has eight attribute values, one for each of the definitions of Section IV-A. The partial traces associated to states {AB} and {AC} (two events each) have 15 attribute values each, since the existence of two activities doubles the number of Occurrence, Cycle, Position, and Distance (one for each activity, 8 in total) and increases to $2^2 = 4$ the number of Duple values, plus other 3 attributes (Change, Single and Elapsed Time). Finally, the partial traces associated to State {ABC} (three events) has 24 attribute values; *i)* Occurrence, Cycle, Position, and Distance produce in this case $4 * 3 = 12$ attributes (one for each activity); whilst *ii)* Duple produces $3^2 = 9$; and finally *iii)* Change, Single and Elapsed Time. These lists are updated when a new case runs. The new traces are inserted in the list that corresponds to the state which

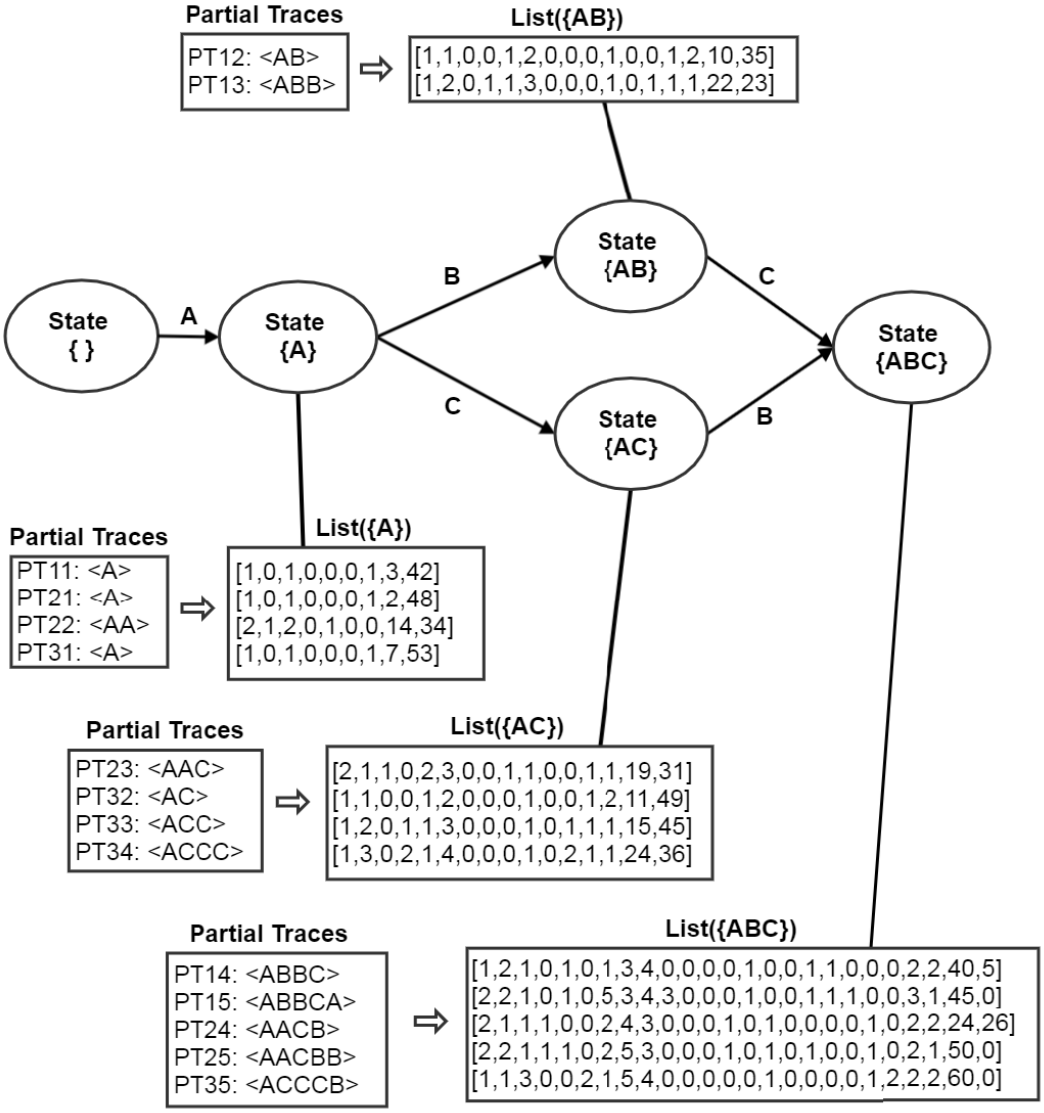

FIGURE 3. Extended annotated transition system with the lists of elements (vectors of attributes values) associated to each state. The example is built using the traces in Table 3, as indicated in Section IV-B.

represents this new trace. Once this is done, the EATS is endowed with all the information needed by our remaining time estimation model. As described in the following sections, estimations will be done with a linear regression model which, for each state, uses as independent variables the values of the attributes in the EATS lists of elements.

C. ESTIMATION OF THE REMAINING TIME: DATASET PARTITIONING

Before describing the details of the regression model we use for estimating the remaining time, we should take into account the following considerations. Let us recall, in the first place, that according to Definition 15, each state S in the EATS is annotated with a list of elements (vectors) $List(S)$ which include the values of the attributes for all the partial traces represented by S. Each of these lists contains all the data (predictors or independent variables) needed for performing the remaining time estimation. Therefore, we have a single dataset associated to each state.

Applying a linear regression technique to each of these datasets has proved to produce poor estimations since, usually, traces in the dataset have great variability in terms of size, number of activities and execution times. This is the usual general scenario for real business process data, such as administrative procedures or applications, industrial incidents management or processes in a hospital or other big organisations/institutions [21], [22]. In many cases, the remaining time values range is vast (e.g. from a few seconds to hundred thousand seconds) even for traces that are very similar or even identical.

A motivational example of this is presented in the real case taken from [21], shown in Table 4. In column "Non-partitioned" we can see the accuracy results after applying a linear regression method on the whole dataset associated

TABLE 4. Accuracy values for the set of partial traces associated to some states taken from a real business process described in [21], following two strategies: Non-partitioned and partitioned.

Accuracy		
States	Non-partitioned	Partitioned
{A}	0.53	0.78
{AB}	0.50	0.77
{ABC}	0.23	0.67
{E}	0.37	0.70
{AE}	0.39	0.73
{ABCF}	0.34	0.69

to each state. We can see that very poor accuracy values are obtained, ranging from 0.23 to 0.53. Since huge variability is a usual feature in real cases we have endowed our estimation method with a dataset partitioning stage, which is described in what follows. In Section IV-D, we will revisit again this motivational example to assess the impact of the partitioning stage on the accuracy results.

Our partitioning method consists of building partitions which contain partial traces with similar remaining times. the similarity is expressed here in terms of a threshold value in the following way: let us assume we have two partial traces PT_1, PT_2, with their corresponding remaining times RT_1, RT_2. Without loss of generality, let us consider that $RT_1 \leq RT_2$. We will consider that PT_1 and PT_2 belong to the same partition if $RT_1/RT_2 > th$, where $th \in [0, 1]$ is a predefined threshold. This condition states that both partial traces belong to the same partitions if the remaining time of the PT_1 is above a given percentage (th) of the PT_2. On the contrary, if $RT_1/RT_2 \leq th$, PT_1 and PT_2 will belong to different partitions. For example, for two traces PT_1 and PT_2 and a threshold value 0.40, this means that if the remaining time of PT_1 is above 40% of PT_2 the will be grouped in the same partition (their remaining times are considered similar). Otherwise (below or equal 40%), they will be grouped in different partitions (their remaining times are considered not similar).

In Algorithm 2, we presented a detailed description of our Threshold-Based Partitioning (TBP) procedure for a given state s and a threshold th. In the first place (line 1) the List associated to s is ordered accordingly to the remaining time of all the partial traces associated to s. Partition building is described in the *for* loop (lines 4-10). In line 6 this condition is formalised as indicated before. When it holds, a new partition is started (lines 7-8). When it does not, the *for* loop in line 4 continues iterating throughout all the partial traces and grouping them in the same partition (line 5). Finally, the last partial trace is assigned (line 11) to its corresponding partition: a new partition in case the threshold condition was met and the last partition in case it did not.

One key issue here is the number of intervals (or threshold points) to be defined (i.e. if we want to create very close intervals or wide intervals). It is important to define the number of segments in a balanced way, not to be very low, since the range values for the estimation time will be high and lower accuracy, but not to be very high, since this will increase the computational cost and will produce over-fitting. In Section V, we will discuss this issue again from a quantitative pragmatical point of view, in order to provide a range of threshold values with a right balance between low accuracy and over-fitting.

Algorithm 2 Threshold-Based Partitioning (TBP) of the Partial Traces (List) Associated to a State s in the EATS

Input: s, a state of the EATS (Def. 14); $th \in [0, 1]$ a partition threshold

Output $PL = \{P_1, .., P_n\}$: Partitions List

1: $SortedList(s) \leftarrow List(s)$ sorted in ascending order by the Remaining Time RT_p of its elements E_p, $p = 1, \ldots, P$ (Def. 14)
2: $n = 1$ ▷ Partition 1
3: $P_n = \varnothing$ ▷ Initialise Partition 1
4: **for** $p = 1, \ldots, P-1$ **do** ▷ P: List size
5: $\quad P_n \leftarrow P_n \cup E_p$
6: $\quad$ **if** $RT_p/RT_{p+1} \leq th$ **then** ▷ Abrupt change: new partition
7: $\quad\quad n \leftarrow n + 1$ ▷ New partition
8: $\quad\quad P_n = \varnothing$ ▷ Initialise the new partition
9: $\quad$ **end if**
10: **end for**
11: $P_n \leftarrow P_n \cup E_p$ ▷ Last Element in the List
12: **return** $PL : \{P_1, \ldots, P_n\}$

In Fig. 4, we show an example of application of the Threshold-based Partitioning Procedure (Algorithm 2 for a threshold value $th = 0.4$. We highlight the quotient values that fulfil the threshold condition (Algorithm 2, line 6) and therefore, define the limits of each partition. All these quotients are less than the 0.4 threshold (respectively 790/4,105 = 0.19 for elements E_6 and E_7, 9024/44,358 = 0.20 for elements E_13 and E_14 and 90,615/232,486 = 0.39 for elements E_16 and E_17), indicating that the corresponding remaining times are not similar and therefore will be grouped in different partitions. After partitioning, this dataset splits into four partitions and, therefore, it is ready for the linear regression model to be applied to each partition, as described in the following section.

D. LINEAR REGRESSION MODEL

Once the need for partitioning the dataset was established and our partitioning strategy was described we are now in conditions to describe our remaining time estimation model. Basically, it is made up of linear regression functions which are obtained for each of the partitions of the dataset described before. The independent variables of the regression functions are the values of attributes (elements) in each partition, being the dependent variable the remaining time.

A simple motivational example showing the impact of this partitioning stage in the remaining-time estimation is shown

	Att_1	Att_2	Att_3	Att_4	....	Att_N	RT	
E_1	1	0	0	2	....	2	255	Partition 1
E_2	1	1	0	1	....	0	316	
E_3	2	0	0	2	....	0	-	
E_4	0	0	1	1	....	2	-	
E_5	1	1	1	1	....	2	-	
E_6	3	2	0	1	....	4	790	
E_7	5	1	2	2	....	0	4105	Partition 2
E_8	2	2	2	2	....	1	4708	
E_9	1	2	3	1	....	0	-	
E_{10}	0	3	1	2	....	0	-	
E_{11}	1	1	0	1	....	0	-	
E_{12}	2	4	0	1	....	1	8055	
E_{13}	1	2	1	1	....	0	9024	
E_{14}	3	1	0	0	....	2	44358	Partition 3
E_{15}	4	0	1	0	....	1	78140	
E_{16}	1	2	1	1	....	2	90615	
E_{17}	4	2	2	1	....	2	232486	Partition 4
E_{18}	2	3	4	3	....	1	243044	
E_{19}	1	1	0	1	....	0	264167	
E_{20}	0	1	1	2	....	0	411337	
E_{21}	0	2	1	2	....	0	696901	
E_{22}	0	0	2	0	....	0	-	
E_{23}	1	2	2	3	....	5	-	
E_{24}	2	3	0	1	....	3	-	
E_{25}	2	3	3	2	....	1	2326398	

FIGURE 4. Example of application of Algorithm 2 for a threshold value *th*=0.4 and a state *S* whose associated list *List*(*S*) is made up by 25 elements. *List*(*S*) is divided into four partitions, which are delimited by elements E_6, E_{13} and E_{16}, as indicated in the text.

in Table 4 (which was introduced in the previous section). Now, in column ''Partitioned'' we present the accuracy results after applying the linear regression method to each of these partitions. Comparing these results with the ones in column ''Non-partitioned'' we can see an important improvement in the accuracy results, which now range from 0.67 to 0.78. On the basis of this example, we present in the next section a detailed validation of our method.

The details of the remaining time estimation for any new trace PT_{NEW} are described in Algorithm 3. Once the state associated to PT_{NEW} is obtained (line 1), its partition list $PL = \{P_1, \ldots, P_n\}$ is returned by Algorithm 2 (line 2). Also, the associated vector of attributes of the new trace is obtained (as indicated in Def. 14) and stored in the corresponding Element. The algorithm basically searches for the Partial Traces in PL, which are the closest ones to PT_{NEW} in terms of the Manhattan distance of the values of their attributes, as indicated in Def. 14. Searching for these Partial Traces and obtaining their associated partitions is made in lines 5-15. Once the Partitions are obtained, their indexes are stored in *partitionIndex* and their corresponding linear regression functions $\{RL_k, k \in partitionIndex\}$ are applied to the values of their attributes of PT_{NEW}. The average of these results is returned as the estimation of the remaining time of our model.

V. VALIDATION AND EXPERIMENTS

In this section, we will describe the experiments we have conducted for validating our proposal. Ten Real-life event

Algorithm 3 Time Prediction Model (TPM) of a new trace

Input: PT_{NEW}: new Partial Trace
Output PRT: Predicted Remaining Time for PT_{NEW}.

1: $S \leftarrow S(PT_{NEW})$ ▷ State which represents PT_{NEW}
2: $PL = \{P_1, \ldots, P_n\}$:= Partitions List associated to S, as returned by Algorithm 2
3: $RL = \{R_1, \ldots, R_n\}$:= List of Linear Regression functions associated to PL, as indicated in Section IV-D
4: E_{NEW} := Element associated to PT_{NEW} ▷ (Def. 14)
5: $distanceMin \leftarrow +\infty$
6: **for** $k = 1, \ldots, n$ **do** ▷ For all partitions in PL
7: **for each** $PT \in P_k$ **do**
8:

$$dist = \sum_{m=1}^{M} |valueOfAtt_m(PT) - valueOfAtt_m(PT_{NEW})|$$

9: **if** $dist < distanceMin$ **then** ▷ New min
10: $partitionIndex = \{k\}$;
11: $distanceMin \leftarrow dist$
12: **else if** $dist == distanceMin$ **then**
13: $partitionIndex = partitionIndex \cup \{k\}$;
14: **end if**
15: **end for**
16: **end for**
17: $PRT \leftarrow$ Average of the estimations obtained with the regression models $\{RL_k, k \in partitionIndex\}$ applied to E_{NEW}
18: **return** PRT

logs were used for validation, as described in Table 5: BPIC12w [21], BPIC13 [23], BPIC15 [24] (5 logs), BPIC17 [22], Hospital Billing [25] and Road Traffic Fine Management Process [26]. Eight of the datasets are taken from the Business Process Intelligence Challenges (BPIC), a *de-facto* standard for testing and validation of business processes approaches, since these datasets are proposed for the yearly Business Process Intelligence Contest.

We have performed three types of experiments with the following aims: *i)* to compare our method, for different threshold values, with the baseline ATS-based proposal [6] in order to validate it in terms of accuracy and mean absolute error and also to have experimental evidence about the influence of the threshold values in the results; *ii)* to determine a single threshold value which could be labelled as the most appropriate choice to use for estimating the remaining time for new event logs, in terms of precision of the results and simplicity of the model; and *iii)* to perform a comparison between our approach and to the sixteen state of the art approaches described in [13] in order to prove the validity of our approach and assess its quality, confronting it to ATS-based, non-ATS based and machine learning approaches.

Following the usual validation methodology, each partition is divided into two parts of 80% training and 20% for testing.

TABLE 5. Description of the 10 real-life event logs used for validation.

Event logs	Description	# Cases	# Events
BPIC12w [21]	Application process for a personal loan or overdraft within a Dutch Financial Institute	8,723	60,780
BPIC13 [23]	Handling Incidents Process from Volvo IT of Belgium.	7,554	57,742
BPIC15 [24] (5 logs)	All building permit applications over four years provided by three Dutch municipalities.	1,199 832 1,409 1,053 1,156	52,217 44,354 59,681 47,293 59,083
BPIC17 [22]	Application process for a personal loan or overdraft within a Dutch Financial Institute	31,509	41,862
Hospital Billing [25]	Billing financial data from the ERP system of a Dutch hospital.	100,000	451,359
Traffic Fine [26]	Data from an information system managing road traffic fines.	150,370	561,470

For each of the partitions, the regression model will produce an estimation for the remaining time following the procedure previously described in Algorithm 3. For testing, for each partial trace, we apply, as indicated in Section IV, the procedure described in Algorithm 3.

A. RESULTS CONSIDERING DIFFERENT THRESHOLD VALUES

In this section, we present the validation results of our approach for different threshold values of the partitioning strategy (TBP) described in Section IV-C. Our method is compared to the ATS baseline approach described in [6] for the ten datasets in Table 5. This validation aims to provide a general overview of the dependence of our remaining time estimation results with the TBP threshold values. We used two metrics to compare our results to [6]: the Mean Absolute Error (MAE) to measure the error between real remaining and the predicted remaining time and the Accuracy to assess the regression quality in objective terms. According to [13], using RMSE as a metric should be avoided in this context, since it is very sensitive to outliers. For this analysis of the threshold dependence, datasets were pre-processed for removing those values that are more/less than twice the standard deviation from the average.

We also consider here the key issue that was previously pointed in section IV-A: in general, a business process could involve a high number of activities, which would also mean that the number of attributes could also be high. As a consequence, also the number of operations related to the partitioning stage would increase. Therefore, in order to keep our approach general, it is advisable to include an attribute selection method that reduces the number of attributes instances initially considered.

In order to do this, we use the well-known Greedy Forward attribute selection method [27] to reduce the number of attributes and, consequently, the complexity of the model as well as to generally improve the scalability of our method. This attribute selection method performs a greedy forward search through the space of attribute subsets, starting with no attributes and stopping when the addition of any remaining attributes results in a decrease in the evaluation. Table 6 shows the difference in the number of attributes considered with and without applying the attribute selection technique. In average, a 31% reduction is achieved in the number of attributes.

TABLE 6. The difference in the number of attributes with and without the Greedy Forward attribute selection technique [27].

Event log	# Activities	# Attributes	
		w/o attribute selection	with attribute selection
BPIC12w	60	546	172
BPIC13	12	91	52
BPIC15_1	280	2,335	846
BPIC15_2	256	1,842	485
BPIC15_3	311	2,508	611
BPIC15_4	275	2,104	407
BPIC15_5	263	1,994	862
BPIC17	107	839	219
Hospital Billing	167	1,566	402
Traffic Fine	24	203	41

In Table 7, we show the results of the comparison between our approach with attributes selection and the base-line approach [6], for the ten considered real-life event logs and the Mean Absolute Error (MAE) metric. We provide the results of our approach for a number of thresholds, in order to assess the general quality of the results as well as the dependence of specific thresholds values. In order to have a general view of the method we also provide (Table 7, bottom row), for each dataset, the average results of our method for all the thresholds considered. According to this, the average value (even including the standard deviation) of our method outperforms [6] in all ten datasets. Looking at all the thresholds values, we can see that for all the 90 cases considered our approach outperforms [6]. The average MAE of our method is 6.76 days, being the average MAE of [6] 43.84 days. The differences in MAE between our approach and [6] range from 2.02 days (in BPIC17) to 126.84 days (in Traffic Fine), being 37.08 days the average of the differences, in all cases favourable to our method.

In Table 8, we show the same comparison for accuracy. Again, the average value (also considering the standard deviation) of our method outperforms [6] in all ten datasets and for the 90 cases considered. For the datasets considered, the average accuracy of our method ranges in [0.54, 0.93],

TABLE 7. Comparison between our approach and [6], using the ten event logs described in Table 5. We show the MAE measurement values at each threshold point.

	Mean Absolute Error (MAE)									
Logs	BPIC12w	BPIC13	BPIC15_1	BPIC15_2	BPIC15_3	BPIC15_4	BPIC15_5	BPIC17	Hospital Bill	Traffic Fine
Aalst et al. [6]	5.69	6.45	42.94	89.68	19.76	20.06	46.83	6.28	52.33	148.36
Threshold										
0	1.398	1.587	3.850	8.738	6.509	4.558	5.956	5.012	13.194	22.465
0.25	1.486	1.629	3.290	9.248	5.540	4.667	5.502	3.370	12.847	20.958
0.5	1.313	1.497	3.070	8.715	6.148	4.228	5.956	3.661	11.806	20.574
0.7	1.695	1.611	3.560	8.495	5.520	4.194	5.787	3.600	14.931	22.911
0.75	1.256	1.520	3.320	9.676	6.121	4.258	5.394	3.879	13.194	19.335
0.9	1.189	1.442	3.000	8.044	5.910	4.166	5.208	4.687	13.310	24.290
0.925	**1.146**	1.381	3.150	7.234	4.954	3.492	4.745	5.671	13.194	20.958
0.95	1.154	**1.134**	3.080	7.778	**4.281**	3.314	4.271	5.557	**11.227**	23.842
0.975	1.256	1.846	**2.070**	**6.296**	4.471	**3.263**	**3.623**	**2.863**	12.153	**18.326**
Average ± STDEV	1.322 ± 0.18	1.516 ± 0.20	3.154 ± 0.49	8.247 ± 1.04	5.495 ± 0.78	4.016 ± 0.53	5.160 ± 0.80	4.255 ± 1.01	12.873 ± 1.07	21.518 ± 2.01

TABLE 8. Comparison between our approach and [6], using the ten event logs described in Table 5. We show the accuracy measurement values at each threshold point.

	Accuracy									
Logs	BPIC12w	BPIC13	BPIC15_1	BPIC15_2	BPIC15_3	BPIC15_4	BPIC15_5	BPIC17	Hospital Bill.	Traffic Fine
Aalst et al. [6]	0.39	0.41	0.44	0.48	0.47	0.55	0.56	0.17	0.35	0.50
Threshold										
0	0.690	0.785	0.894	0.898	0.893	0.921	0.925	**0.563**	0.649	0.782
0.25	0.641	0.774	0.896	0.899	0.906	0.921	0.928	0.535	0.682	0.785
0.5	0.690	0.770	0.905	0.897	0.900	0.918	0.925	0.558	0.700	0.797
0.7	0.705	0.790	0.898	0.904	0.899	0.918	0.930	0.549	0.693	0.775
0.75	0.718	0.771	0.899	0.899	0.901	0.924	0.935	0.562	0.710	0.753
0.9	0.691	0.800	0.908	0.902	0.908	0.927	0.936	0.546	**0.730**	0.784
0.925	**0.756**	0.791	0.912	0.912	0.907	0.933	0.938	0.491	0.705	0.785
0.95	0.738	0.835	0.909	0.915	0.916	0.936	0.941	0.526	0.723	0.772
0.975	0.694	**0.843**	**0.920**	**0.929**	**0.923**	**0.940**	**0.950**	0.533	0.720	**0.806**
Average ± STDEV	0.703 ± 0.03	0.795 ± 0.03	0.905 ± 0.01	0.906 ± 0.01	0.906 ± 0.01	0.927 ± 0.01	0.934 ± 0.01	0.540 ± 0.02	0.701 ± 0.03	0.782 ± 0.02

whilst [6] ranges in [0.17, 0.56]. The average accuracy of our method is 0.81, being the average accuracy of [6] 0.43.

B. THRESHOLD CHOICE

In this section, we discuss how to choose an appropriate threshold value which could be used, in general, for any new dataset. We will support the discussion with the experimental analysis and results we describe in what follows. In a first analysis, it seems straightforward that the best threshold choice should be the most precise one, i.e., that produces the highest accuracy or lowest MAE. In order to experimentally determine which is, in general, the most precise threshold, we need to consider the results in Tables 7 and 8, rank them, and calculate the average rank through all the datasets and thresholds, for both MAE and Accuracy. These results are summarised in Table 9. According to this, the most precise threshold (MPT) is the one with the lowest ranks, which is 0.975 for both Accuracy and MAE.

TABLE 9. The average rank, and the standard deviation through all the real event logs.

	Accuracy		MAE	
Threshold	Avg rank	STDEV	Avg rank	STDEV
0	6.9	2.4	7.1	1.4
0.25	6.5	1.7	6.0	2.3
0.5	6.3	2.4	5.0	2.1
0.7	5.9	1.5	6.3	2.1
0.75	5.4	2.1	5.6	2.0
0.9	4.0	1.4	4.9	2.0
0.925	3.7	2.2	3.7	2.3
0.95	3.3	2.5	3.1	2.5
0.975	**2.2**	**2.1**	**2.4**	**2.5**

In a second analysis, we can observe looking at the results in Tables 7 and 8 that, in general, high threshold values produce better results (lower MAE and higher accuracy).

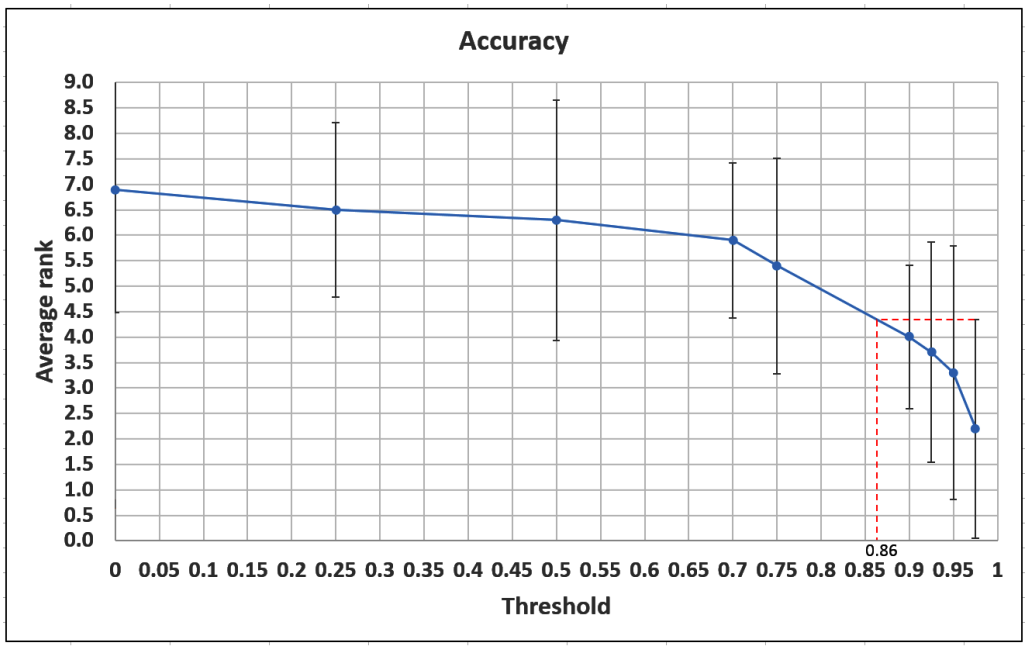

FIGURE 5. Model selection using the one-standard-error rule method [28] for the accuracy results in Table 9. The selected model corresponds to a 0.86 threshold.

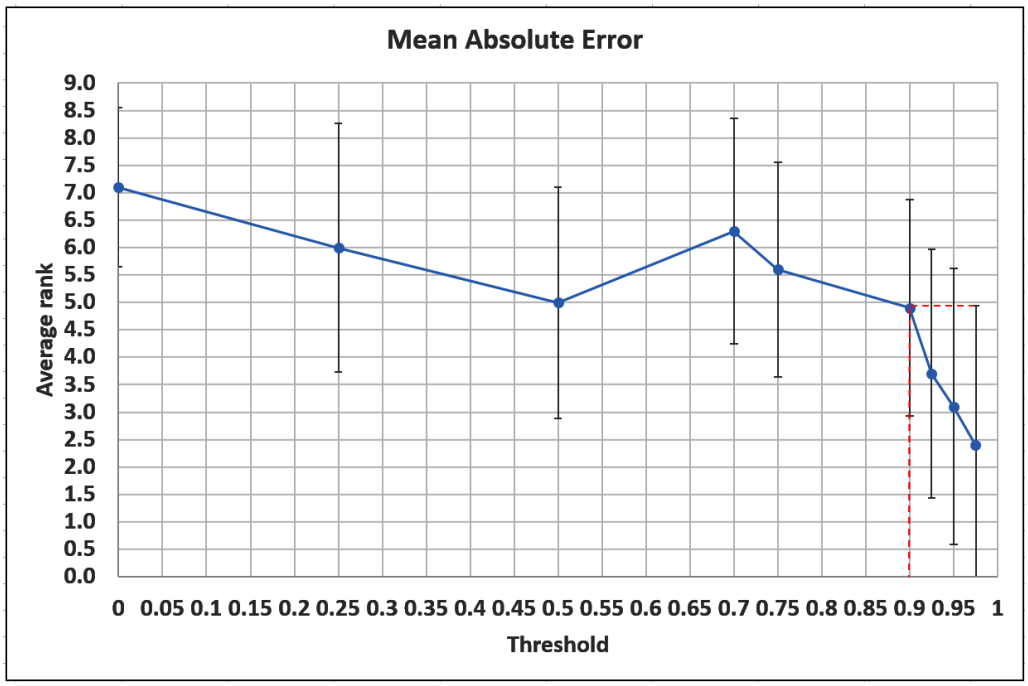

FIGURE 6. Model selection using the one-standard-error rule method [28] for the MAE results in Table 9. The selected model corresponds to a 0.90 threshold.

This means that the granularity of the partitioning intervals is higher, and as a consequence, the number of operations involved also increases.

Taking also this consideration into account, it becomes evident that, in the general case, choosing the most appropriate threshold is not necessarily only a matter of choosing the most precise one, but the one with a good balance between precision in the results (generally associated to high thresholds) and the number of operations involved (generally associated to low thresholds). In order to determine a threshold value with a good compromise between these two opposite constraints, we will apply the One-Standard-Error Rule [28], which is a well-known model selection technique. Through applying this rule, together with the attribute

TABLE 10. MAE results (in days) of the 16 approaches compiled in [13] compared to our approach in two different scenarios (BCT and MPT, in the first two rows). Symbol "-" Xin some cells means this value is not provided by the authors.

Technique	BPIC2012w	BPIC2013	BPIC2015_1	BPIC2015_2	BPIC2015_3	BPIC2015_4
BCT	2.39±2.81	**7.14±6.92**	4.00±18.99	6.67±23.47	1.28±7.39	4.37±21.16
MPT	**1.90±2.39**	7.28+7.78	**3.74±13.13**	**6.02±26.27**	**1.27±8.53**	**2.79±14.37**
TS	7.505 ± 1.036	-	56.498 ± 8.341	118.293 ± 16.819	26.412 ± 8.082	61.630 ± 5.413
LSTM	6.344 ± 0.994	-	39.457 ± 5.708	61.620 ± 2.061	19.682 ± 2.646	48.902 ± 1.527
SPN	8.538 ± 0.772	-	66.509 ± 17.131	81.114 ± 8.033	26.757 ± 10.378	51.202 ± 5.889
FA	6.946 ± 1.057	-	-	-	-	-
cluster_agg	7.180 ± 0.953	-	40.705 ± 1.824	68.185 ± 2.649	23.087 ± 3.226	51.555 ± 2.363
cluster_index	7.074 ± 1.254	-	38.092 ± 2.988	66.957 ± 3.436	24.497 ± 1.887	56.113 ± 6.411
cluster_last	7.061 ± 1.019	-	38.388 ± 3.478	62.781 ± 2.347	22.544 ± 1.656	51.451 ± 4.189
prefix_agg	7.260 ± 0.935	-	46.765 ± 23.581	71.210 ± 8.893	24.152 ± 2.785	53.568 ± 6.413
prefix_index	7.155 ± 0.942	-	37.525 ± 2.746	66.883 ± 3.756	21.861 ± 3.292	50.452 ± 4.605
prefix_last	7.139 ± 0.851	-	37.975 ± 5.903	64.708 ± 5.749	23.574 ± 3.778	53.053 ± 5.665
noBucket_agg	7.082 ± 1.020	-	35.962 ± 3.744	67.914 ± 2.467	24.453 ± 3.577	54.89 ± 1.894
noBucket_index	6.982 ± 1.340	-	35.451 ± 2.499	65.505 ± 3.442	23.025 ± 1.587	52.282 ± 1.182
noBucket_last	7.021 ± 1.099	-	37.442 ± 3.607	64.110 ± 2.332	25.150 ± 1.271	56.818 ± 1.729
state_agg	7.465 ± 0.622	-	42.949 ± 2.725	68.768 ± 4.094	28.427 ± 9.844	49.318 ± 2.699
state_index	7.510 ± 0.585	-	-	-	-	-
state_last	7.539 ± 0.554	-	42.946 ± 2.691	68.296 ± 3.762	27.826 ± 8.280	49.038 ± 2.498

Technique	BPIC2015_5	BPIC2017	Hospital Bill	Traffic Fine
BCT	2.87±15.10	3.42±3.07	13.67±21.82	43.99±62.41
MPT	**2.23±12.64**	**3.27±2.84**	**12.85±20.69**	**33.31±58.35**
TS	67.699 ± 7.531	8.278 ± 2.468	46.491 ± 21.344	190.949 ± 15.447
LSTM	52.405 ± 3.819	7.150 ± 2.635	36.258 ± 23.870	178.738 ± 89.019
SPN	-	10.731 ± 0.370	71.377 ± 29.082	193.807 ± 96.796
FA	-	-	51.689 ± 14.945	223.808 ± 14.859
cluster_agg	45.825 ± 3.028	7.479 ± 2.282	42.934 ± 26.136	210.322 ± 98.516
cluster_index	44.587 ± 4.378	-	-	209.139 ± 98.417
cluster_last	46.433 ± 4.085	7.457 ± 2.359	48.589 ± 26.708	208.599 ± 99.549
prefix_agg	46.396 ± 2.466	7.525 ± 2.306	43.06 ± 25.884	201.614 ± 99.484
prefix_index	44.290 ± 3.669	7.421 ± 2.360	41.698 ± 25.944	209.085 ± 99.708
prefix_last	46.639 ± 3.718	7.482 ± 2.325	48.528 ± 26.714	209.304 ± 102.027
noBucket_agg	49.203 ± 1.833	7.437 ± 2.381	43.483 ± 25.0	211.017 ± 93.198
noBucket_index	50.153 ± 1.097	-	-	208.879 ± 92.250
noBucket_last	49.027 ± 1.954	7.525 ± 2.244	50.496 ± 23.961	204.758 ± 93.399
state_agg	49.873 ± 2.658	-	43.835 ± 25.984	211.439 ± 98.351
state_index	-	-	41.095 ± 26.499	210.408 ± 99.276
state_last	49.556 ± 2.575	7.521 ± 2.341	48.902 ± 27.001	209.206 ± 100.632

selection technique described in Section IV-A, we will provide experimental support for selecting a threshold value with a balanced compromise between precision and number of operations involved and that can, therefore, be labelled also as an appropriate choice for any new dataset.

The One-Standard-Error Rule is applied to the ranking described in Table 9 and looks for the lowest threshold value whose average error is no more than one standard deviation above the error of the best model. The threshold value obtained following this procedure (BCT, Best Compromise Threshold) will exhibit a good compromise between precision and the number of operations involved.

Figures 5 and 6 show the application of the One-Standard-Error Rule for both the accuracy and MAE results in Table 9. For both figures, the blue line indicates the average ranking values for each threshold considered. The red line indicates the rank value that corresponds to the best model in terms of average rank plus one standard deviation. The intersection of both lines indicates the Best Compromise Thresholds, respectively 0.86 for Accuracy (Figure 5) and 0.90 for MAE

(Figure 6) which are the recommended thresholds in terms of precision and number of operations involved.

C. COMPARISON TO OTHER APPROACHES

In Table 10, we show the comparison results for the MAE metric between our proposal and other 16 states of the art methods describe in a very recent survey [13], for the same ten datasets reported in the previous sections. The comparison was made under the same conditions as in [13], in order to obtain comparable results. Such conditions are: *(i)* no dataset pre-processing was made, *(ii)* the prefix length of the traces was 20 activities (i.e., the last 20 activities were considered for the remaining time estimation), and *(iii)* MAE is the metric used to assess the quality of the remaining time estimations.

For a fair and more detailed comparison, we provide the results of our method in two scenarios for the threshold selection: MPT provides the results obtained for the Most Precise Threshold (0.975) and BCT for the MAE Best Compromise Threshold (0.90) as defined in Section V-B. It can be seen that our approach, even in the worst cases, produces the lowest error in all the datasets. Compared with the best model reported in [13] (LSTM [18]), which is a deep learning based approach, the average MAE of our MPT method is 7.49 days, being the average MAE of LSTM 50.02 days. Differences in MAE between our approach and LSTM range from 3.88 days (BPIC2017) to 145.07 days (Traffic Fine), being 38.28 days the difference average favourable to our method.

We have also considered the impact of the standard deviation, which is higher in our method than the others for most of the datasets considered. Comparing the worst case (MAE+STDEV), the average of our method is 35.59 days, being 58.25 days the average of LSTM. Differences between our approach and LSTM range with this metric from 3.05 days (BPIC2012w) to 175.74 days (Traffic Fine), being 35.59 days the difference average, in all cases favourable to our method. According to these results, our method still outperforms LSTM when considering variance.

Apart of these experimental results, interpret-ability is also a key advantage of our method, when compared to Deep Learning approaches, which are usually labelled as black-box approaches. Since our EATS approach is based on linear regression on attribute values that are related to the structure and contents of the trace, users can interpret and understand the variables meaning and their relative importance (coefficients in the regression expressions). It should be taken into account that interpret-ability of systems is an increasing demand in the context of Fairness, Accountability, Transparency, and Ethics (FATE) in Artificial Intelligence and systems and applications in general.

VI. CONCLUSION AND FUTURE WORK

In this paper, we proposed a new approach for predicting the remaining time of the running process in business process management. Our approach consists of two perspectives: firstly, we define a number of attributes that are evaluated from the process traces and capture quantitative and structural information about them. Secondly, a linear regression model is used for remaining time prediction, using these attributes. The attributes are added to the well-known annotated transition system (ATS, [6]), thus producing a new Extended ATS which takes into account structural information of the traces. Furthermore, to deal with the trace variability in terms of size, the number of activities and execution times, a threshold-based partitioning method of the dataset logs is proposed. For each of the partitions, a different linear regression model is obtained. The evaluation of our approach was made using ten real-life event logs, showing that our model outperforms the results in the state of the art [13], particularly the LSTM Deep Learning approach [18], in terms of Mean Absolute Error and Accuracy metrics. The scalability of our approach has been addressed by considering and validating an attribute selection method and a model selection method for obtaining a single threshold value that can be recommended as an appropriate choice for new estimation problems.

The main limitations of our proposal are: *(i)* the proposed attributes may not be able to capture all the structural richness in some scenarios; *(ii)* the partitioning method could be further improved for adapting better to other cases; *(iii)* linear regression estimators for remaining time will not capture non-linear behaviours which could exist in some complex cases; and, finally, *(iv)* for complex cases involving a huge number of activities other scalability approaches could be needed apart of the model selection approach we have described. In this regard, as current and future work, we are considering new attributes definitions as well as other partitioning and regression methods. New model simplification approaches will also be proposed and compared with the current proposal, so that we could further enhance the good remaining time estimation results we have presented in this paper.

REFERENCES

[1] A. Rogge-Solti and M. Weske, "Prediction of business process durations using non-Markovian stochastic Petri nets," *Inf. Syst.*, vol. 54, pp. 1–14, Dec. 2015. doi: 10.1016/j.is.2015.04.004.

[2] B. F. van Dongen, R. A. Crooy, and W. M. P. van der Aalst, "Cycle time prediction: When will this case finally be finished?" in *Proc. Int. Conf. Move Meaningful Internet Syst. (OTM)*, vol. 5331, R. Meersman and Z. Tari, Eds. Berlin, Germany: Springer, 2008, pp. 319–336. doi: 10.1007/978-3-540-88871-0_22.

[3] W. M. P. van der Aalst, *Process Mining: Data Science in Action*, 2nd ed. Berlin, Germany: Springer, 2016. doi: 10.1007/978-3-662-49851-4.

[4] S. Pandey, S. Nepal, and S. Chen, "A test-bed for the evaluation of business process prediction techniques," in *Proc. 7th Int. Conf. Collaborative Comput. (COLLABORATECOM)*, D. Georgakopoulos and J. B. D. Joshi, Eds., 2011, pp. 382–391. doi: 10.4108/icst.collaboratecom.2011.247129.

[5] M. Polato, A. Sperduti, A. Burattin, and M. de Leoni, "Data-aware remaining time prediction of business process instances," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2014, pp. 816–823. doi: 10.1109/IJCNN.2014.6889360.

[6] 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. doi: 10.1016/j.is.2010.09.001.

[7] W. M. P. van der Aalst and S. Dustdar, "Process mining put into context," *IEEE Internet Comput.*, vol. 16, no. 1, pp. 82–86, Jan./Feb. 2012. doi: 10.1109/MIC.2012.12.

[8] W. van der Aalst, "Process mining: Overview and opportunities," *ACM Trans. Manage. Inf. Syst.*, vol. 3, no. 2, pp. 7.1–7.17, 2012. doi: 10.1145/2229156.2229157.
[9] W. van der Aalst *et al.*, "Process mining manifesto," in *Business Process Management Workshops* (Lecture Notes in Business Information Processing), vol. 99, F. Daniel, K. Barkaoui, and S. Dustdar, Eds. Heidelberg, Germany: Springer-Verlag, 2012, pp. 169–194. doi: 10.1007/978-3-642-28108-2_19.
[10] A. Bolt and M. Sepúlveda, "Process remaining time prediction using query catalogs," in *Business Process Management Workshops (BPM)*, vol. 171, N. Lohmann, M. Song, and P. Wohed, Eds. Cham, Switzerland: Springer, 2014, pp. 54–65. doi: 10.1007/978-3-319-06257-0_5.
[11] C. D. Francescomarino, C. Ghidini, F. M. Maggi, and F. Milani, "Predictive process monitoring methods: Which one suits me best?" in *Proc. 16th Int. Conf. Bus. Process Manage. (BPM)*, vol. 11080, M. Weske, M. Montali, I. Weber, and J. vom Brocke, Eds. Cham, Switzerland: Springer, 2018, pp. 462–479. doi: 10.1007/978-3-319-98648-7_27.
[12] 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. doi: 10.1109/TSC.2017.2772256.
[13] I. Verenich, M. Dumas, M. L. Rosa, F. M. Maggi, and I. Teinemaa, "Survey and cross-benchmark comparison of remaining time prediction methods in business process monitoring," 2018, *arXiv:1805.02896*. [Online]. Available: https://arxiv.org/abs/1805.02896
[14] W. M. P. van der Aalst, M. Pesic, and M. Song, "Beyond process mining: From the past to present and future," in *Proc. Int. Conf. Adv. Inf. Syst. Eng. (CAiSE)*, vol. 6051, B. Pernici, Ed. Berlin, Germany: Springer, 2010, pp. 38–52. doi: 10.1007/978-3-642-13094-6_5.
[15] M. Ceci, P. F. Lanotte, F. Fumarola, D. P. Cavallo, and D. Malerba, "Completion time and next activity prediction of processes using sequential pattern mining," in *Proc. 17th Int. Conf. Discovery Sci. (DS)*, vol. 8777, S. Džeroski, P. Panov, D. Kocev, and L. Todorovski, Eds. Cham, Switzerland: Springer, 2014, pp. 49–61. doi: 10.1007/978-3-319-11812-3_5.
[16] I. Verenich, H. Nguyen, M. L. Rosa, and M. Dumas, "White-box prediction of process performance indicators via flow analysis," in *Proc. Int. Conf. Softw. Syst. Process (ICSSP)*, R. Bendraou, D. Raffo, L. Huang, and F. M. Maggi, Eds., 2017, pp. 85–94. [Online]. Available: http://dl.acm.org/citation.cfm?id=3084100
[17] C. D. Francescomarino, M. Dumas, F. M. Maggi, and I. Teinemaa, "Clustering-based predictive process monitoring," *IEEE Trans. Services Comput.*, to be published. doi: 10.1109/TSC.2016.2645153.
[18] N. Tax, I. Verenich, M. La Rosa, and M. Dumas, "Predictive business process monitoring with LSTM neural networks," in *Proc. CAiSE*, vol. 10253, E. Dubois and K. Pohl, Eds. Essen, Germany: Springer, 2017, pp. 477–492. doi: 10.1007/978-3-319-59536-8_30.
[19] E. Cesario, F. Folino, M. Guarascio, and L. Pontieri, "A cloud-based prediction framework for analyzing business process performances," in *Proc. Int. Conf. Availability, Rel., Security (CD-ARES)*, vol. 9817, F. Buccafurri, A. Holzinger, P. Kieseberg, A. M. Tjoa, and E. Weippl, Eds. Cham, Switzerland: Springer, 2016, pp. 63–80. doi: 10.1007/978-3-319-45507-5_5.
[20] A. Senderovich, C. Di Francescomarino, C. Ghidini, K. Jorbina, and F. M. Maggi, "Intra and inter-case features in predictive process monitoring: A tale of two dimensions," in *Proc. 15th Int. Conf. Bus. Process Manage. (BPM)*, vol. 10445, J. Carmona, G. Engels, and A. Kumar, Eds. Cham, Switzerland: Springer, 2017, pp. 306–323. doi: 10.1007/978-3-319-65000-5_18.
[21] B. F. Van Dongen. (2012). *BPI Challenge 2012*. [Online]. Available: https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f
[22] B. F. Van Dongen. (2017). *BPI Challenge 2017*. [Online]. Available: https://doi.org/10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b
[23] W. Steeman. (2014). *BPI Challenge 2013*. [Online]. Available: https://doi.org/10.4121/uuid:a7ce5c55-03a7-4583-b855-98b86e1a2b07
[24] B. F. Van Dongen. (2015). *BPI Challenge 2015*. [Online]. Available: https://doi.org/10.4121/uuid:31a308ef-c844-48da-948c-305d167a0ec1
[25] F. Mannhardt. (2016). *Hospital Billing—Event Log*. [Online]. Available: https://doi.org/10.4121/uuid:76c46b83-c930-4798-a1c9-4be94dfeb741
[26] F. Mannhardt and M. de Leoni. (2014). *Road Traffic Fine Management Process*. [Online]. Available: https://doi.org/10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5
[27] R. Caruana and D. Freitag, "Greedy attribute selection," in *Machine Learning Proceedings*, W. W. Cohen and H. Hirsh, Eds. San Mateo, CA, USA: Morgan Kaufmann, 1994, pp. 28–36. doi: 10.1016/B978-1-55860-335-6.50012-X.
[28] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*, vol. 112. New York, NY, USA: Springer, 2013. doi: 10.1007/978-1-4614-7138-7.

AHMAD ABUROMMAN received the B.Sc. degree in information technology from Al-Balqa Applied University (BAU), Jordan, in 2005, and the M.Sc. degree in information technology (e-Services) from Griffith University, Australia, in 2009. He is currently pursuing the Ph.D. degree in time prediction in running business processes with the Research Center in Intelligent Technologies, University of Santiago de Compostela (CiTIUS).

MANUEL LAMA received the Ph.D. degree in physics from the University of Santiago de Compostela, in 2000, where he is currently an Associate Professor of artificial intelligence. He has collaborated on more than 30 projects and research contracts financed by public calls, participating as a principal investigator in 20 of them. These activities were implemented in areas, such as process discovery, predictive monitoring, and management of dynamic processes. As a result of this research, he has published over 150 scientific articles with review process in conference and national and international journals.

ALBERTO BUGARÍN is currently a Full Professor of artificial intelligence and a Principal Researcher with the Intelligent Systems Group (GSI), Research Centre in Intelligent Technologies, University of Santiago de Compostela (CiTIUS). He is a coauthor of almost 200 scientific articles on these subjects and has participated in more than 40 research and development projects and contracts for innovation and transfer of technology, being a Principal Researcher in 18 of them. He has also participated in scientific dissemination projects. His research interests include artificial intelligence and its applications, such as natural language generation (data-to-text systems), soft computing, machine learning, and approximate knowledge representation and reasoning.

He is a member of several thematic research networks. He is currently a Board Member of the Spanish Association for Artificial Intelligence (AEPIA) and an Organizing Co-Chair of the 24th European Conference on Artificial Intelligence (ECAI) (Santiago de Compostela, in 2020).