Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2021) 29: 122 – 137
© TÜBİTAK
doi:10.3906/elk-2005-71

Time-oriented interactive process miner: a new approach for time prediction

İsmail YÜREK[1,*], **Derya BİRANT**[2], **Özlem Ece YÜREK**[1], **Kökten Ulaş BİRANT**[2]
[1]The Graduate School of Natural and Applied Sciences, Dokuz Eylül University, İzmir, Turkey
[2]Department of Computer Engineering, Faculty of Engineering, Dokuz Eylül University, İzmir, Turkey

Received: 11.05.2020 • **Accepted/Published Online:** 10.09.2020 • **Final Version:** 27.01.2021

Abstract: Everyday information systems collect a different kind of process instances of a business flow. As time goes on, the size of the collected data builds up speedily and constitutes a huge amount of data. It is a very challenging task to obtain valuable information and features of processes from such big data. Considering in advance, the trend and different features of the ongoing process are essential. Especially, time management is crucial in designing and conducting business processes. In this article, a novel process miner algorithm is proposed for time prediction, named time-oriented İnteractive process miner (T-IPM), which predicts the remaining and completion time of each process in a business workflow. The goal is to develop a process miner algorithm for time prediction that is able to work on a huge amount of event logs, and to incorporate the execution records of ongoing processes into the discovered process model instantly. In this study, three datasets from different domains that include event logs of repair, hospital, and traffic process flows were used to carry out the experimental studies in order to show the effectiveness of the proposed time prediction algorithm. The experimental studies show that the proposed algorithm is capable of performing process analysis on a huge amount of event logs with low memory consumption and high prediction accuracy.

Key words: Process mining, information systems, workflow management, information processing and management, prediction methods

1. Introduction

Processes exist in several areas of life. We are involved in many different processes in our daily lives and business lives. These processes are recorded by the existing information systems that contain knowledge about the activities carried out. The recorded execution traces are called as event logs. The event log contains different types of important information about the processes. Process mining is a technical way to acquire information from event logs. Event logs containing elaborative materials about the history of business operations are analyzed under several aspects to analyze, discover, improve, and manage the business process with the help of developed algorithms in process mining.

Everyday an information system captures business operations; therefore, the volume of data that event logs contain increases rapidly. It is a challenging task to extract valuable knowledge from this kind of massive amount of data. In some process flows, it is significant to predict with up-to-date data. In this case, it is crucial to analyze the execution records of ongoing processes instantaneously. That means we need to update the time prediction model in online fashion.

Event logs contain varied crucial features of the processes such as name, cost, resource, location, times-

[*]Correspondence: ismail.yurek@gmail.com

tamp, etc. Time is one of the most important features of processes. Considering time is vital in understanding, designing/redesigning, operating, and managing business processes, time management has great importance in controlling the lifecycle of processes. It provides continuous improvement in business processes. To ensure the improvement of business processes, process engineers, supervisors, quality managers, and managers need to know about the business processes' execution time in advance. Time prediction algorithms provide insight into the execution durations of ongoing processes. In addition, it helps to detect bottlenecks of processes and to take the proper actions about situations.

In light of the above information, it is clearly seen that time prediction is a very important task in process mining. In this paper, we mainly focus on the issue of the remaining and completion time of processes. One of the most essential features of the proposed algorithm is that it quickly and instantly adds the completion and execution time information of newly completed or ongoing processes into the time prediction model. The algorithm calculates the time-related aspects of the already-executed portion of a business process flow and puts the result of calculations to analyze data to make new predictions based on it.

In this study, we aim at finding the answers to these questions: When an ongoing process instance will end? What is the average completion time of activity A? What is the average execution time of activity B? What is the remaining time of an ongoing process instance? What is the needed time to handle a customer's claim? We proposed a process miner algorithm for time prediction to find answers to these questions. The experimental studies were carried out in the repair[1], hospital[2], and traffic[3] datasets to evaluate the accuracy of the developed algorithm and observe its performance.

The main novelty of this paper is that it proposes a new process miner algorithm for time prediction, named time-oriented interactive process miner (T-IPM), which efficiently analyzes event logs related to business workflow and predicts both the remaining and completion time of each process in a workflow. The other main contributions of this article are as following:

- Developing a time prediction algorithm capable of performing process analysis on a huge amount of event logs with low memory consumption.
- Handling the execution records of ongoing processes to make a more accurate prediction.

The remainder of the article is structured as follows: Section 2 briefly explains the current and related literature as well as the previous works on the subject. Section 3 describes the proposed algorithm, T-IPM, and explains how to construct a time prediction model from event logs. In Section 4, the experimental study is presented and the obtained experimental results are discussed. Section 5 presents concluding remarks as well as possible future works.

2. Related works

At the end of the 90s, the process mining concept began to become popular. A new approach was presented by Agrawal et al. [1] which copes with parallel structure and noise to expand the utility of actual workflow systems. Using this approach, the user can apply existing event logs to model a given business process as a graph. Cook and Wolf [2] stated different techniques for process discovery and generated formal models based on the actual process executions.

[1] Van der Aalst WMP (2008). Repair Dataset [online]. Website http://www.processmining.org/logs/ [accessed 9 May 2020].

[2] Mannhardt F (2017). Hospital Billing - Event Log Dataset [online]. Website https://research.tue.nl/en/datasets/hospital-billing-event-log [accessed 9 May 2020].

[3] De Leoni M, Mannhardt F (2015). Road Traffic Fine Management Process Dataset [online]. Website https://research.tue.nl/en/datasets/road-traffic-fine-management-process [accessed 9 May 2020].

Eder et al. [3] emphasized the importance of time management in workflow-based process management systems. They proposed a framework to compute the deadlines of activities to see that all time constraints are successfully satisfied and the end-to-end process deadline is met. They presented ways to check time constraints like lower and upper bound between activities at process build and process instantiation time.

As one of the most known process mining algorithms, the α-algorithm was introduced by van der Aalst et al. [4] to discover a large and relevant class of workflow processes. This algorithm analyzes the event log and then constructs various dependency relations between tasks. The goal of the α-algorithm is to analyze different kinds of workflow logs in the presence of noise without any knowledge of the underlying process. Cook and Wolf [5] used probabilistic methods during the study about finding concurrent models of system behavior from event logs. Herbst and Karagiannis [6] interested in duplicate tasks and they presented an algorithm based on an inductive manner in two steps: (i) induction and (ii) stochastic task graph (SAG) generation. The SAG is converted to a blocked structured model using a definition language. Schimm [7] offered an approach to get an accurate model from event logs. This approach deals with hierarchically structured workflow models that include the splits and joins.

Dongen and van der Aalst [8] introduced a data model and an XML format called MXML (ining eXtensible markup language) to specify a standard for storing event logs. Eder and Pichler [9] worked on the notion of probabilistic time management to improve the predictions about the remaining duration of a given workflow. It is stated that different routes may be selected in a workflow, so taking into consideration each path's probabilities is important. They introduced the probabilistic timed graph, which shows the time histograms and branching probabilities.

Weijters et al. [10] came up with an algorithm, called Heuristics Miner, which discovers the main behavior registered in a noisy event log by including different threshold parameters in order to overcome two problems: noise and low-frequency behavior. Reijers [11] made a point of case prediction challenges and defined the case prediction difficulties as part of business process management systems. This paper minds the forecasting of the remaining time to complete a specific case.

Günther and van der Aalst [12] focused on existing problems in the traditional process mining methods when the processes are large and less-structured. In order to manage the problems, they built up a flexible approach based on their previous works [3, 13]: fuzzy mining. Their approach analyzes, simplifies, and visualizes mined process models based on the metrics about the significance and correlation of graph elements. Medeiros et al. [14] tried to mine process models by using a genetic algorithm in the ProM framework. They performed experiments on the simulated dataset. The results of the experiments showed that the genetic algorithm found all possible business process models. However, the main disadvantage of the genetic approach is time and space complexity. For this reason, a distributed genetic approach was proposed by Bratosin et al. [15] to overcome the high computational problem of the genetic approach.

Dongen et al. [16] centered on the remaining time of process flow. They calculated the remaining time by using the regression method. This paper stated that nonparametric regression is very appropriate when no or very limited precedents are present. By applying this method, predicting the cycle times in any uncertain case in a business process is possible. They took into account the duration and occurrence of all activities and showed that their approach does better from taking the average time minus the already passed time with a real-life example. Verwer et al. [17] defined an algorithm that depends on the state combining method for learning a deterministic finite-state machine. The algorithm is used for learning a time-based model from observations. Schonenberg et al. [18] suggested an efficient recommendation service. This service is able to work with flexible and modular process-aware information systems (PAISs). It supports users while process execution is proceeding by giving advising about possible next stages. They created these recommendations which were given by the service depending on similar past process executions. Also, they considered specific optimization goals while studying different methods to calculate log-based recommendations.

Leitner et al. [19] presented an approach to predict the service level agreement (SLA) violations. Measured and predicted facts are used as input in order to create a prediction model. The paper stated that regression methods create the prediction model, and historical process examples are used during the training phase. The user can determine the machine learning method that will be applied by defining an algorithm and selecting its respective parameters.

Van der Aalst et al. [20] studied on process mining for operational decision making. They suggested a framework for time-based operational support and developed it in the ProM tool. This study shows that process mining techniques are not only limited to past processes but can also be utilized for both current and future processes. It is said that existing process mining algorithms consider the historical information, but in this study van der Aalst et al. are interested in individual process instances that are still running and incomplete.

Van der Aalst et al. [21] provided a configurable approach to predict the completion time of the process by constructing a process model with time information. In the paper, it is stated that they seriously focus on the transition system generation. They used this annotated transition system to estimate all or some of the business process examples' remaining execution time. It is presented that the algorithm they proposed to predict the completion of a case performs better than simple heuristic scanning and also performs better than regression models in terms of precision and efficiency. They also highlighted that their method can be improved to estimate other features of a case such as the time until a particular event or the occurrence of a particular event by annotating the transition system with related information/additional features.

Van der Aalst [22] presented basic types of process mining (PM), enhancement, discovery, and conformance checking, and emphasized PM as one of the popular subjects in business process management (BPM). He illustrated the applicability of process mining types in real-life examples. Van der Aalst and his team developed process mining tools: EMiT (enhanced mining tool), little thumb, and ProM (process mining). EMiT can describe process flows with Petri nets [23]. Little thumb can process models from noisy and incomplete event logs [13]. ProM is an open-source tool to develop process mining projects that include many packages with many plug-ins [24].

Fahland and van der Aalst [25] simplified discovered process models and demonstrated a post-processing approach to check the balance between under-fitting and overfitting. The constructed process model is detailed by using a Petri net. Their approach can be combined with any process discovery method which generates Petri net. Appice et al. [26] worked on a process mining method and especially used predictive clustering to prepare an execution flow. This model expresses the last events of running cases to forecast the features of coming events. They used a predictive clustering tree (PCT) to predict ongoing events. They implemented their approach in the ProM framework and explained its verification and effectiveness with several case studies.

Polato et al. [27] presented a novel method to enhance the quality of prediction by building an effective process model that is annotated with time and information to predict the remaining time. The paper stated that the estimation of the remaining time is made by combining the likelihood of all the executed events. They considered the data attributes' values to predict the remaining time of an ongoing case.

The comparison of different process mining methods was given by Aleem et al. [28] in detail. The paper collects and shows all efficient and qualitative results of business process mining for researchers. The process mining approaches are grouped into five sections in the article. The sections are defined as: deterministic, heuristic, inductive, genetic, and clustering-based mining approaches. Cheng and Kumar [29] worked on noisy traces. They suggested a method to eliminate noisy traces from event logs. The method follows the way of building a classifier and applying classifier rules on event logs. They indicated that the preprocessed event logs produced superior mined process models on several evaluation metrics. The problem of repairing the discovered process model was investigated to align them to reality by Fahland and van der Aalst [30]. The event log was separated into different sub logs of nonfitting traces to make conformance checking. Rovani et al. [31] worked on medical treatment processes. They demonstrated a methodology to analyze the processes in medical treatment operations. They displayed how to apply process mining methods based on the declarative model.

De Leoni at al. [32] suggested a framework for correlation analysis. They worked to unify several approaches. In their paper, the different process characteristics were tried to be correlated from a different perspective. Mannhardt et al. [33] proposed a process mining algorithm for conformance checking. They tried to control process conformance based on control flow, data dependencies, resource alignment, and time constraints. Pika et al. [34] suggested an approach to analyze event logs. They analyzed the information of process executions recorded in event logs. A supporting tool was demonstrated to forecast process outcomes and to assess the overall risk of process. Tax et al. [35] proposed an algorithm named the local process model. They worked to discover the most frequent behavioral flows in event logs. The main focus area of the algorithm is local structures. The algorithm tries to enable the process mining of noisy event logs and extend sequential pattern mining techniques. A framework was built by Bolt et al. [36] to ensure that process mining is quotable and automated for event logs that might need to be separated and distributed for analysis. Various analysis cases are explained to demonstrate the feasibility of their work. Polato et al. [37] offered three new prediction methods to estimate the remaining time of ongoing cases. They took into consideration the extended data in the event log in addition to the process flow information. They used machine learning methods so as to build models that can deal with additional information. In the paper, it is explained that the proposed approach is able to cope with unexpected behaviors or noisy data. The suggested algorithms were evaluated on real-life data and they showed the performance of algorithms.

Suriadi et al. [38] worked on noisy event logs. They specified a set of data quality issues. They suggested a patterns-based way to clean noisy event logs. Mitsyuk et al. [39] tried to produce event logs. They proposed a tool to create event logs from the business process model and notation (BPMN). They used script-based gateways and choice preferences to manage the control flow. Bolt et al. [40] addressed the problem between execution behaviors and business rules. They also dealt with the different variants of the same process. They annotated the transitions systems with measurements. The annotated transition systems were used to model behavior and to show differences.

Alizadeh et al. [41] recommended an approach to enable the identification of deviations. They linked data and control flow for conformance checking. Their study stated that the proposed approach explains the purpose and context of the deviations identified.

Yürek et al. [42] proposed a novel algorithm, named interactive process miner (IPM), that has the capabilities of working on a huge amount of event logs and managing the execution flows of ongoing process instances to generate process model with high precision and fitness values. Also, IPM supports a simulation environment for users.

In this article, a novel process miner algorithm is introduced for time prediction by enhancing the IPM algorithm [42] with considering time perspective. The enhanced algorithm, time-oriented interactive process miner (T-IPM), is capable of predicting the remaining and completion time of each process in a business workflow. Differently from the previous studies, the proposed algorithm can work on a huge amount of event logs with low memory consumption and incorporate the execution records of ongoing processes into the discovered process model instantly.

3. The proposed approach: time-oriented interactive process miner (T-IPM)

This section of the article will explain how to construct a time prediction model from event logs. The proposed algorithm, T-IPM, is able to analyze both offline and online execution records. Incorporation of ongoing or newly completed process records, which are called online data, has major importance in keeping the process model constantly up-to-date. Updating the process model instantly will provide the ability to see the latest status of business workflows. The main objective of this paper is to introduce a process miner algorithm for time prediction with an up-to-date process model with the ability to work on a huge amount of event logs with low memory consumption.

Event logs contain different features of executed process instances. In the scope of this study, activity, timestamp, and lifecycle information is important for the proposed algorithm. Event logs are stored in the XES (eXtensible event stream) file format. The proposed algorithm reads the log files by using the file streaming method, so it is possible to read the event logs without loading the whole file into memory. Thus, the size of log files is not essential for memory allocation.

The first step is to create a tree-like data model that includes summarized statistical information about all cases in event logs. This data model describes the summary information that will be used in time prediction. The tree-like data model makes it possible to use a huge amount of event logs in time prediction algorithm with low memory consumption. T-IPM algorithm is developed based on IPM [42] algorithm. It is an enhanced version of the IPM algorithm by adding time perspective. Due to the nature of the proposed algorithm, there is no need to recreate the prediction model by analyzing all event logs. The proposed algorithm is able to analyze historical event logs as well as to incorporate the execution records of ongoing processes into the process model instantly. In this way, the algorithm can update the trained model gradually by adding new event logs.

The constructed data model contains all the activities in the event log. Including all cases and activities of each case in an event, the log makes the data model complex and hard to process. Also, an event log may contain noise or may include incomplete flows.

We consider the frequency of cases in the event log to eliminate the effects of incompleteness and noise. Also, we get the data model more readable to make it easy to understand and process.

Cases are eliminated from the data model with a defined threshold. This threshold is a parametric value, and it can take a value between 0 and 1. For instance, if the threshold value is set to 0.25, it means that the cases which have the frequency value under % 25 are eliminated.

A sample event log is listed in Table 1. The event log contains cases that have activity, timestamp, and lifecycle information. The lifecycle features hold information about the process status. The sample log contains three activities such as A, B, and C. For example, in Case 1, activity A starts at 12:00 and ends at 18:00. The execution of activity A takes 6 h. Activity B starts at 20:00 and ends at 11:00 the next day. The execution of activity B takes 15 h. There is idle time for 2 h between the end of activity A and the start of activity B.

Figure 1 shows the tree-like data model that is constructed from the sample event log. We have three

Table 1. The event log.

Case Id	Activity	Timestamp	Lifecycle
1	A	2018-01-01T12:00	Start
1	A	2018-01-01T18:00	Complete
1	B	2018-01-01T20:00	Start
1	B	2018-01-02T11:00	Complete
1	C	2018-01-02T12:30	Complete
1	D	2018-01-02T14:00	Start
1	D	2018-01-02T15:30	Complete
1	F	2018-01-02T16:00	Complete
2	A	2018-01-03T09:00	Complete
2	B	2018-01-03T11:00	Start
2	B	2018-01-03T15:30	Complete
2	D	2018-01-03T16:00	Start
2	D	2018-01-03T17:00	Complete
2	F	2018-01-03T18:30	Complete
3	A	2018-01-04T11:00	Start
3	A	2018-01-04T11:30	Complete
3	C	2018-01-04T14:00	Complete
3	D	2018-01-04T15:00	Start
3	D	2018-01-04T15:30	Complete
3	F	2018-01-04T15:51	Complete

different cases in the event log as ABCDF, ABDF, and ACDF. It means that process instances follow three different paths from start to end. These paths are shown in Figure 1 as *L0*, *L1*, and *L2*.

The relation between the activities is indicated by an edge in Figure 1. An edge is a link between two process activities and indicates a dependency between these activities as such the activity A is executed after the activity B. Conceptually, an incoming edge is an edge containing an expectation that another activity occurs in execution before it is started or continued. On the other hand, an outcoming edge is termed as an edge containing an expectation that another activity will occur in execution after it is ended. When we follow the *L0* path, we recognize that four edges are used (*E0*, *E1*, *E2*, and *E3*). Activity A has two outgoing edges (*E0* and *E6*). Activity B has one incoming edge (*E0*) and two outgoing edges (*E1* and *E4*). Activity C exists in two different paths. It has two incoming edges (*E1* and *E6*) and two outgoing edges (*E2* and *E7*). Activity D is located in three different paths. Activity D has three incoming edges (*E2*, *E4*, and *E7*) and three outgoing edges (*E3*, *E5*, and *E8*). Activity F has only incoming edges (*E3*, *E5*, and *E8*).

The proposed algorithm calculates each activity's execution and completion times in process flows to make a prediction. *Execution time* indicates the time difference between the start time and end time of activity. If there is no start or end time information in the event log, we assume that the execution time value is zero. *Completion time* refers to the time difference between the end time of the previous activity and the next activity's end time.

Figure 2 shows the difference between the execution and the completion times of activity. If the completion

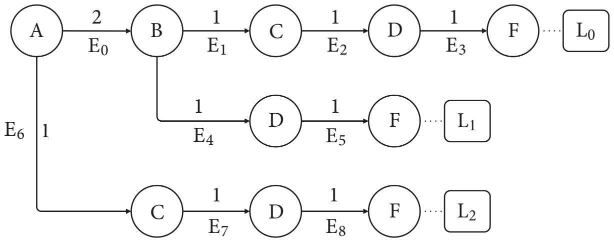

Figure 1. The data model of event log.

time value is greater than the execution time value, it means there is an idle time between two activities. If the completion time value is smaller than the execution time value, this means these two activities progress in parallel.

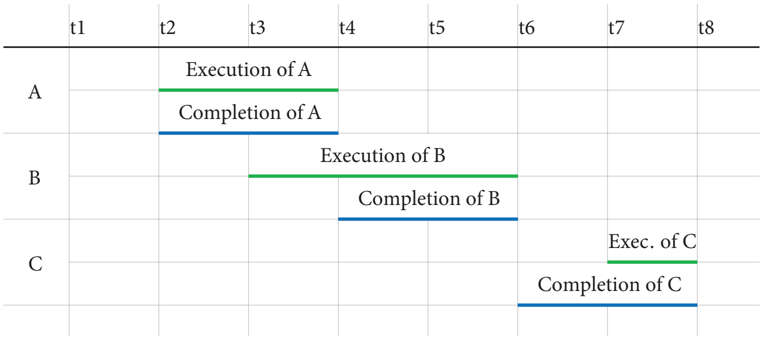

Figure 2. The timeline of execution and completion times.

A sample process flow that continues like A → B → C is shown in Figure 2. The activity A starts at t_2 and ends at t_4. The activity B starts at t_3 and ends at t_6. The activities A and B continue parallel for a certain period of time. The activity C starts at t_7 and ends at t_8. After the activity B is completed, there is an idle time which starts at t_6 and ends at t_7. After the idle time finishes, the activity C starts to proceed.

To explain the time prediction, we need the following notations. Z is a set that contains a number of activities such as $Z = \{$A, B, C, D, E$\}$. $\sigma \in$ Z* is a case of process flow such as $\sigma_1 =$ ABC, $\sigma_2 =$ ABCD. $W \subseteq Z*$ is an event log, i.e., a multiset (bag) of event cases such as $W =$ [ABC, ABCD, ABDE].

Let $a \in$ Z; $T_{start}(a)$ is the start time of a. $T_{end}(a)$ is the end time of a. $T_{et}(a)$ is the execution time of a, and $T_{ct}(a)$ is the completion time of a.

If there is a case $\sigma = e_1 e_2 e_3 ... e_n$ and i $\in$ {1, 2, 3,..., n}, where n is the total number of activities in the case σ, such that $e_i = a$, then *execution time* is defined by equation (1) and *completion time* is defined by equation (2).

$$T_{et}(a) = T_{end}(a) - T_{start}(a) \tag{1}$$

$$T_{ct}(a) = \begin{cases} T_{end}(a) - T_{end}(e_{i-1}), i > 1 \\ \\ T_{et}(a), i = 1 \end{cases} \tag{2}$$

Equation (1) is used to calculate the execution time and equation (2) is used to calculate the completion time of an event. By using equation (1) and equation (2), it is possible to calculate the execution and completion times of each activity in a process flow. In the next step, we will see how the average values are calculated.

Let us assume that $a \xrightarrow{\text{in}} b$ is the count of incoming edges from a to b and $a^{\overset{in}{\to}}$ is the count of all incoming edges into activity a. Equation (3) shows how to calculate the average execution time of A → B transition and equation (4) shows how to calculate the average completion time of A → B transition. The average values are firstly calculated for 2-length activities such as A → B. If there is no relation between activities, we need to calculate the average values by using 1-length activities. In this case, all incoming edges are taken into account to compute average values.

$$AVG_{et}(a,\ b) = \begin{cases} \frac{\Sigma_{i=1}^{|a \xrightarrow{\text{in}} b|} T_{et}(b)}{|a \xrightarrow{\text{in}} b|}, |a \xrightarrow{\text{in}} b| > 0 \\ \\ \frac{\Sigma_{i=1}^{|b \xrightarrow{\text{in}}|} T_{et}(b)}{|b \xrightarrow{\text{in}}|}, |a \xrightarrow{\text{in}} b| = 0 \end{cases} \tag{3}$$

$$AVG_{ct}(a,\ b) = \begin{cases} \frac{\Sigma_{i=1}^{|a \xrightarrow{\text{in}} b|} T_{ct}(b)}{|a \xrightarrow{\text{in}} b|}, |a \xrightarrow{\text{in}} b| > 0 \\ \\ \frac{\Sigma_{i=1}^{|b \xrightarrow{\text{in}}|} T_{ct}(b)}{|b \xrightarrow{\text{in}}|}, |a \xrightarrow{\text{in}} b| = 0 \end{cases} \tag{4}$$

So far, the formulas of execution time, completion time, and average time values are defined. Now, the prediction formulas can be defined as follows. Equation (5) defines the prediction of execution time and equation (6) defines the prediction of completion time.

$$P_{et}(\sigma) = AVG_{et}(\varnothing, e_0) + \Sigma_{i=1}^{n} AVG_{et}(e_i, e_{i+1}) \tag{5}$$

$$P_{ct}(\sigma) = AVG_{ct}(\varnothing, e_0) + \Sigma_{i=1}^{n} AVG_{ct}(e_i, e_{i+1}) \tag{6}$$

To apply prediction formulas, we need time tables that show the average execution and completion times. The data model that is shown in Figure 2 is used as input to create the time tables. The algorithm starts from the root activity and continues traversing the tree to form time tables. The goal is to reach the leaf activity while traversing the tree. In our example, there are three paths as L_0, L_1, and L_2 to reach the leaf activities. All paths are traversed when all leaf activities are reached; it means that the constructed tree represents the complete event log because it represents all cases in the event log which are visited so far.

A time table is created for each 1-length and 2-length activities while traversing all the data model activities. The average execution and completion times of each relation are calculated while traversing the data model. Table 2 gives time values of 2-length activities and Table 3 gives time values of 1-length activities.

Let us explain the time prediction algorithm with a sample flow. Assume that a flow of σ_1 = ACBDF will take place. This flow is transformed into binary transitions to estimate the execution and completion times. At the end of this operation, binary transitions are obtained such as ∅ → A, A → C, C → B, B → D, and D → F.

Table 2. The time information of 2-length activities.

Transition	#	Total execution time (min.)	Average execution time (min.)	Total completion time (min.)	Average completion time (min.)
∅ → A	3	390	130	390	130
A → B	2	1170	585	1410	705
A → C	1	0	0	150	150
B → C	1	0	0	90	90
B → D	1	60	60	90	90
C → D	2	120	60	270	135
D → F	3	0	0	141	47

Table 3. The time information of 1-length activities.

Transition	#	Total execution time (min.)	Average execution time (min.)	Total completion time (min.)	Average completion time (min.)
A	3	390	130	390	130
B	2	1170	585	1410	705
C	2	0	0	240	120
D	3	180	60	360	120
F	3	0	0	141	47

Next, it is necessary to find the average time for each binary transition to calculate the completion time with equation (6). Table 4 displays the calculated average time values. If there is a transition in the event log such as $a \rightarrow b$, the calculated average value for this pair is obtained from Table 2. If there is no such transition, the overall calculated average value for activity b is obtained from Table 3.

The average completion time values for each transition pair in the example flow are shown in Table 4. There is no average completion value for C → B transition (2-length). The average completion value of activity B is used for estimation as 705 (1-length). These values are summed to estimate the completion time value for the corresponding flow. Equation (6) is calculated as 1122 min for σ_1.

Table 4. The average completion time of transitions.

Transition	2-length (min.)	1-length (min.)	Average (min.)
∅ → A	130	130	130
A → C	150	120	150
C → B	-	705	705
B → D	90	120	90
D → F	47	47	47

Equation (5) is used to calculate the execution time. The average execution time values for each binary transition in the sample flow are presented in Table 5. There is no average execution value for C → B transition (2-length). The average execution value of activity B is used for estimation as 585 (1-length). The estimated execution time for σ_1 is calculated as 645 minutes by taking into account the average time values.

Table 5. The average execution time of activities.

Transition	2-length (min.)	1-length (min.)	Average (min.)
$\varnothing \rightarrow$ A	0	0	0
A $\rightarrow$ C	0	0	0
C $\rightarrow$ B	-	585	585
B $\rightarrow$ D	60	60	60
D $\rightarrow$ F	0	0	0

4. Experimental study

Three experiments were performed using a 2.4 GHz quad-core processor, 16GB RAM to evaluate the proposed algorithm, T-IPM. The code was implemented in the Java platform.

The processes in the three datasets, repair, hospital, and traffic, were used to test the proposed algorithm. These datasets were enlarged 250 times in order to observe the effects of the event log size on both running time and memory consumption. In this way, the repair-big, hospital-big, and traffic-big datasets were created by repeating all the cases in the original datasets. Table 6 shows detailed information about the datasets.

Table 6. The characteristics of experimental datasets.

Dataset	Cases	Events	Activities	Mean completion time (hour)
Repair	1104	7733	8	1.08
Hospital	100000	451359	18	3056.68
Traffic	150370	561470	11	8200.29
Repair-big	276000	1933250	8	1.08
Hospital-big	25000000	112839750	18	3056.68
Traffic-big	37592500	140367500	11	8200.29

The repair dataset consists of synthetically created event logs for the telephone repair process. It consists of 1104 cases and 7733 events. The repair-big dataset, which consists of 276000 cases and 1933250 events was generated based on the repair dataset. The hospital dataset that includes a real-life event log was obtained from the information system of a hospital located in the Netherlands. It consists of 100000 cases and 451359 events. The hospital-big dataset, which consists of 25000000 cases and 112839750 events was generated based on the hospital dataset. The traffic dataset includes event logs generated by an information system that performs road traffic control. It consists of 150370 cases and 561470 events. The traffic-big dataset, which consists of 37592500 cases and 140367500 events was generated based on the traffic dataset.

The first experiment was executed on the hospital, traffic, repair, hospital-big, traffic-big, and repair-big datasets to compare the memory consumption and running time. The results of the experiment given in Table 7 show the memory consumption and running time of the proposed algorithm for each dataset.

The proposed algorithm created the process model in 2 s by using 96 MB RAM on the repair dataset and 153 s by using 101 MB RAM on the repair-big dataset. The second performance test was completed in 21 s

Table 7. The experimental results of performance evaluation.

Dataset	Log size (MB)	Running time (sec.)	Memory consumption (MB)
Repair	3.31	2	96
Repair-big	829.27	153	101
Hospital	166.22	21	151
Hospital-big	41525.74	4630	157
Traffic	176.79	25	148
Traffic-big	44180.20	5955	162

by using 151 MB RAM on the hospital dataset and 4630 s by using 157 MB RAM on the hospital-big dataset. The last one was completed in 25 s by using 148 MB RAM on the traffic dataset and 5955 s by using 162 MB RAM on the traffic-big dataset. When we checked the running times and memory consumption, we observed that the event logs' size increased the running time, but it did not increase memory consumption dramatically. Although the sizes of the datasets increased by 250 times, the algorithm consumed up to 9% more memory. The factor that affects the amount of memory consumption is the complexity of the event logs and the length of each process instance. These two cases are the two most important factors affecting the data structure's size in which summary information is obtained from event logs. The experimental results of performance evaluation show that the proposed algorithm is capable of working on a huge amount of event logs with low memory consumption.

The second experiment was executed on the hospital, traffic, and repair datasets to evaluate the time prediction algorithm's success. Table 8 presents the results of the experiment.

Table 8. The experimental results of time prediction.

Dataset	MAE	RMSE	MAPE (%)
Repair	0.1774	0.2370	16.44
Hospital	391.6208	544.8783	12.80
Traffic	1620.5391	1966.9926	19.38

When building the time prediction model, 80% of each dataset was used for training, and the remaining 20% of each dataset was used for testing the model. The error is calculated by the difference between the actual value in the event log and estimated value by the T-IPM algorithm. Table 8 shows *mean absolute error (MEA)*, *root mean squared error (RMSE)*, and *mean absolute percentage error (MAPE)* of the proposed algorithm for each dataset.

MAE is the average of the absolute differences between actual observation and prediction when all individual differences have equal weight. Without considering the direction of values, the average magnitude of the errors in a set of predictions is measured by MAE. MAE is defined by equation (7) as follows;

$$MAE = \frac{1}{n}\Sigma_{i=1}^{n}|x_i - x_i'| \tag{7}$$

RMSE is a quadratic scoring rule that gets square root of the average of squared differences between actual observation and prediction. RMSE measures the average magnitude of the error. MAPE calculates the average of the percentage error. The size of the error is measured in percentage terms. It is defined by equation (8) as follows;

$$MAPE = (\frac{1}{n}\Sigma_{i=1}^{n}\frac{|x_i - x_i'|}{x_i}) \times 100 \tag{8}$$

Table 6 shows the average completion times of the execution records in the repair, hospital and traffic datasets as 1.08 h, 3056.68 h and 8200.29 h, respectively. According to these values, it can be said that a repair process takes an average of 68 min, a hospital process an average of 127 days, and a traffic process an average of 341 days.

Since time values are calculated in hours during analysis, error results should be evaluated in hours. In Table 8, the experimental results of time prediction in the repair dataset show that the MAE value was calculated as 0.1774 h. When this value is converted to minutes and compared with the average completion time, an error of 10.64 min (16.41%) is observed. When the same evaluation is made for other results, it is seen that the error is 391.62 h (12.81%) and 1620.53 h (19.76%) for the hospital and traffic data sets, respectively. These error rates are at a reasonable level for the process mining problem. The effectiveness of the method is basically provided by the proposed tree-like data model that includes summarized statistical information about all cases in event logs. This data model makes it possible to correctly analyze a huge amount of event log data with low memory consumption.

It is more convenient to take into account the MAPE value to explain the accuracy of the proposed algorithm as a percentage. The MAPE value was calculated by averaging the observed error percentages for each prediction. MAPE shows the percentage error of time prediction results. The accuracy values of the time prediction were observed for the repair, hospital and traffic datasets as 83.56%, 87.20%, and 80.62%, respectively. The experimental results of time prediction confirm that the proposed algorithm has high prediction accuracy. The obtained accuracy values indicate the effectiveness of the algorithm since they are higher than an acceptable level (>80%) for the process mining domain. The efficiency of the algorithm is mainly achieved as a result of considering the frequency of cases in the event log to eliminate the effects of incompleteness and noise. The elimination of the redundant cases from the data model with a certain threshold significantly increases the success rate of the algorithm.

5. Conclusion and future work

This article proposes a novel process miner algorithm for time prediction, time-oriented interactive process miner (T-IPM), which is an enhanced version of the IPM algorithm by introducing a time perspective. The proposed algorithm is capable of predicting the remaining time of ongoing processes instantly and the completion time of the processes that have not started yet.

The analysis of event logs is increasingly limited by memory and speed constraints since a huge amount of event logs are collected through information systems. This study's empirical results clearly show that the proposed algorithm, T-IPM, can work on a massive amount of event logs with low memory consumption and handle the execution records of ongoing processes.

The contribution of the proposed algorithm can be summarized as the following:

- Predicting the remaining and completion time of each process in a business workflow.
- Working on huge amount of event logs with low memory consumption.
- Handling the execution records of ongoing processes.

In the future, it is possible to enhance the proposed algorithm by introducing new process mining perspectives such as organizational and resource. A new file format that takes less space can be defined for event logs. Thus, low-cost storage spaces for event logs will suffice. Currently, event logs are stored in the XES file format. Sometimes, the element tags of file format may get more storage space than the execution records of processes.

References

[1] Agrawal R, Gunopulos D, Leymann F. Mining process models from workflow logs. In: Proceedings of the 6th International Conference on Extending Database Technology; Valencia, Spain; 1998. pp. 467-483. doi: 10.1007/BFb0101003

[2] Cook JE, Wolf AL. Discovering models of software processes from event-based data. ACM Transactions on Software Engineering and Methodology 1998; 7 (3): 215-249. doi: 10.1145/287000.287001

[3] Eder J, Panagos E, Rabinovich M. Time constraints in workflow systems. In: Proceedings of the 11th Conference on Advanced Information Systems Engineering (CAiSE); Heidelberg, Germany; 1999. pp. 165-280. doi: 10.1007/3-540-48738-7_22

[4] Van der Aalst WMP, Weijters T, Maruster L. Workflow mining: discovering process models from event logs. IEEE Transactions on Knowledge and Data Engineering 2004; 16 (9): 1128-1142. doi: 10.1109/TKDE.2004.47

[5] Cook JE, Du Z, Liu C, Wolf AL. Discovering models of behavior for concurrent workflows. Computers in Industry 2004; 53 (3): 297-319.

[6] Herbst J, Karagiannis D. Workflow mining with InWoLvE. Computers in Industry 2004; 53 (3): 45-264. doi: 10.1016/j.compind.2003.10.002

[7] Schimm G. Mining exact models of concurrent workflows. Computers in Industry 2004; 53 (3): 265-281. doi: 10.1016/j.compind.2003.10.003

[8] Van Dongen BF, Van der Aalst WMP. A Meta model for process mining data. In: Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAiSE) Workshops; Porto, Portugal; 2005. pp. 309-320.

[9] Eder J, Pichler H. Probabilistic calculation of execution intervals for workflows. In: Proceedings of the 12th International Symposium on Temporal Representation and Reasoning (TIME'05); Burlington, VT, USA; 2005. pp. 183-185. doi: 10.1109/TIME.2005.29

[10] Weijters AJMM, Van der Aalst WMP, De Medeiros AKA. Process Mining with the Heuristics Miner Algorithm. Eindhoven, Netherlands: Eindhoven University of Technology, 2006.

[11] Reijers HA. Case prediction in BPM systems: a research challenge. Journal Of the Korean Institute of Industrial Engineers 2006; 33 (1): 1-10.

[12] Günther CW, Van der Aalst WMP. Fuzzy Mining: Adaptive process simplification based on multi-perspective metrics. In: Proceedings of the 5th International Conference on Business Process Management; Brisbane, Australia; 2007. pp. 328-343. doi: 10.1007/978-3-540-75183-0_24

[13] Weijters AJMM, Van der Aalst WMP. Rediscovering workflow models from event-based data using little thumb. Integrated Computer-Aided Engineering 2003; 10 (2): 151-162. doi: 10.3233/ICA-2003-10205

[14] De Medeiros AKA, Weijters AJMM, Van der Aalst WMP. Genetic process mining: an experimental evaluation. Data Mining and Knowledge Discovery 2007; 14: 245-304. doi: 10.1007/s10618-006-0061-7

[15] Bratosin C, Sidorova N, Van der Aalst WMP. Distributed genetic process mining. In: Proceedings of IEEE Congress on Evolutionary Computation; Barcelona, Spain; 2010. pp. 1-8. doi: 10.1109/CEC.2010.5586250

[16] Van Dongen BF, Crooy RA, Van der Aalst WMP. Cycle time prediction: when will this case finally be finished? In: Proceedings of OTM Confederated International Conferences on the Move to Meaningful Internet Systems; Monterrey, Mexico; 2008. pp. 319-336.

[17] Verwer S, De Weerdt MM, Witteveen C. Efficiently learning timed models from observations. In: Proceedings of Benelearn Conference; Spa, Belgium; 2008. pp. 75-76.

[18] Schonenberg H, Weber B, Van Dongen B, Van der Aalst WMP. Supporting flexible processes through recommendations based on history. In: Proceedings of the 6th International Conference on Business Process Management; Milan, Italy; 2008. pp. 51-66. doi: 10.1007/978-3-540-85758-7_7

[19] Leitner P, Wetzstein B, Rosenberg F, Michlmayr A, Dustdar S et al. Runtime prediction of service level agreement violations for composite services. In: Proceedings of Service-Oriented Computing ServiceWave Workshops; Stockholm, Sweden; 2009. pp. 176-186. doi: 10.1007/978-3-642-16132-2_17

[20] Van der Aalst WMP, Pesic M, Song M. Beyond process mining from the past to present and future. In: Proceedings of the 22nd International Conference on Advanced Information Systems Engineering (CAiSE); Hammamet, Tunisia; 2010. pp. 38-52. doi: 10.1007/978-3-642-13094-6_5

[21] Van der Aalst WMP, Schonenberg MH, Song M. Time prediction based on process mining. Information Systems 2011; 36 (2): 450-475. doi: 10.1016/j.is.2010.09.001

[22] Van der Aalst WMP. Process mining: overview and opportunities. ACM Transactions on Management Information Systems 2012; 3 (2): 1-17. doi: 10.1145/2229156.2229157

[23] Van Dongen BF, Van der Aalst WMP. EMiT: A process mining tool. In: Proceedings of the 25th International Conference on Application and Theory of Petri Nets; Bologna, Italy; 2004. pp. 454-463. doi: 10.1007/978-3-540-27793-4_26

[24] Van Dongen BF, De Medeiros AKA, Verbeek HMW, Weijters AJMM, Van der Aalst WMP. The ProM framework: a new era in process mining tool support. In: Proceedings of the 26th International Conference on Application and Theory of Petri Nets; Miami, FL, USA; 2005. pp. 444-454. doi: 10.1007/11494744_25

[25] Fahland D, Van der Aalst WMP. Simplifying discovered process models in a controlled manner. Information Systems 2013; 38 (4): 585-605. doi: 10.1016/j.is.2012.07.004

[26] Appice A, Pravilovic S, Malerba D. Process mining to forecast the future of running cases. In: Proceedings of the 2nd International Workshop on New Frontiers in Mining Complex Patterns; Prague, Czech Republic; 2013. pp. 67-81. doi: 10.1007/978-3-319-08407-7_5

[27] Polato M, Sperduti A, Burattin A, De Leoni M. Data-aware remaining time prediction of business process instances. In: Proceedings of International Joint Conference on Neural Networks (IJCNN); Beijing, China; 2014. pp. 38-52. doi: 10.1109/IJCNN.2014.6889360

[28] Aleem S, Capretz LF, Ahmed F. Business process mining approaches: a relative comparison. International Journal of Science, Technology & Management 2015; 4 (1): 1557-1564.

[29] Cheng HJ, Kumar A. Process mining on noisy logs - can log sanitization help to improve performance? Decision Support Systems 2015; 79: 138-149. doi: 10.1016/j.dss.2015.08.003

[30] Fahland D, Van der Aalst WMP. Model repair - aligning process models to reality. Information Systems 2015; 47: 220-243. doi: 10.1016/j.is.2013.12.007

[31] Rovani M, Maggi FM, De Leoni M, Van der Aalst WMP. Declarative process mining in healthcare. Expert Systems with Applications 2015; 42 (23): 9236-9251. doi: 10.1016/j.eswa.2015.07.040

[32] De Leoni M, Van der Aalst WMP, Dees MA. A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs. Information Systems 2016; 56: 235-257. doi: 10.1016/j.is.2015.07.003

[33] Mannhardt F, De Leoni M, Reijers HA, Van der Aalst WMP. Balanced multi-perspective checking of process conformance. Computing 2016; 98 (4): 407-437. doi: 10.1007/s00607-015-0441-1

[34] Pika A, Van der Aalst WMP, Wynn WT, Fidge CJ, Ter Hofstede AHM. Evaluating and predicting overall process risk using event logs. Information Sciences 2016; 352-353: 98-120. doi: 10.1016/j.ins.2016.03.003

[35] Taxa N, Sidorovaa N, Haakmab R, Van der Aalst WMP. Mining local process models. Journal of Innovation in Digital Ecosystems 2016; 3 (2): 183-196. doi: 10.1016/j.jides.2016.11.001

[36] Bolt A, De Leoni M, Van der Aalst WMP. Scientific workflows for process mining: building blocks, scenarios, and implementation. International Journal on Software Tools for Technology Transfer 2016; 18 (6): 607-628. doi: 10.1007/s10009-015-0399-5

[37] Polato M, Sperduti A, Burattin A, De Leoni M. Time and activity sequence prediction of business process instances. Computing 2016; 100: 1005-1031. doi: 10.1007/s00607-018-0593-x

[38] Suriadi S, Andrewsa R, Ter Hofstedea AHM, Wynna MT. Event log imperfection patterns for process mining: towards a systematic approach to cleaning event logs. Information Systems 2017; 64: 132-150. doi: 10.1016/j.is.2016.07.011

[39] Mitsyuk AA, Shugurov IS, Kalenkova AA, Van der Aalst WMP. Generating event logs for high-level process models. Simulation Modelling Practice and Theory 2017; 74: 1-16. doi: 10.1016/j.simpat.2017.01.003

[40] Bolt A, De Leoni M, Van der Aalst WMP. Process variant comparison: Using event logs to detect differences in behavior and business rules. Information Systems 2018; 74: 53-66. doi: 10.1016/j.is.2017.12.006

[41] Alizadeh M, Lu X, Fahland D, Zannone N, Van der Aalst WMP. Linking data and process perspectives for conformance analysis. Computers & Security 2018; 73: 172-193. doi: 10.1016/j.cose.2017.10.010

[42] Yürek I, Birant D, Birant KU. Interactive process miner: a new approach for process mining. Turkish Journal of Electrical Engineering & Computer Sciences 2018; 26 (3): 1314-1328. doi: 10.3906/elk-1708-112