Chinese Journal of Electronics
Vol.32, No.3, May 2023

Explainable Business Process Remaining Time Prediction Using Reachability Graph

CAO Rui[1], ZENG Qingtian[1], NI Weijian[1], LU Faming[1], LIU Cong[2], and DUAN Hua[1]

(1. *College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266510, China*)

(2. *School of Computer Science and Technology, Shandong University of Technology, Zibo 255000, China*)

Abstract — With the recent advances in the field of deep learning, an increasing number of deep neural networks have been applied to business process prediction tasks, remaining time prediction, to obtain more accurate predictive results. However, existing time prediction methods based on deep learning have poor interpretability, an explainable business process remaining time prediction method is proposed using reachability graph, which consists of prediction model construction and visualization. For prediction models, a Petri net is mined and the reachability graph is constructed to obtain the transition occurrence vector. Then, prefixes and corresponding suffixes are generated to cluster into different transition partitions according to transition occurrence vector. Next, the bidirectional recurrent neural network with attention is applied to each transition partition to encode the prefixes, and the deep transfer learning between different transition partitions is performed. For the visualization of prediction models, the evaluation values are added to the sub-processes of a Petri net to realize the visualization of the prediction models. Finally, the proposed method is validated by publicly available event logs.

Key words — Explainable predictive process monitoring, Remaining time prediction, Reachability graph, Recurrent neural network, Petri net.

I. Introduction

Predictive process monitoring [1], [2] is one of the most interesting research topics in the area of process mining [3]. It aims to predict the future states of a running process instance, such as outcome prediction [4], next activity prediction [5], [6], and remaining time prediction.

Different machine learning methods, and more recently deep learning algorithms, have been used to predict business process remaining time. A large number of studies [7] show that methods using deep learning, e.g., long short-term memory (LSTM), are more accurate for remaining time prediction. However, remaining time prediction methods based on deep learning suffer from poor interpretability. For example, an encoder-decoder architecture is proposed in [8], which is based on the generative adversarial network. An encoder-decoder architecture for generative adversarial networks that generates a sequence of activities and their timestamps in an end-to-end manner. In [9], a data-driven simulation technique and multiple deep learning techniques are developed, which construct models are capable of generating execution traces with timestamped events. In [10], a method for predictive business process monitoring using time-aware long and short-term memory is proposed. In [11], a method of recurrent neural network with a long and short-term memory structure is proposed to predict the remaining time of business processes. In [12], a method of transfer learning for remaining time prediction is proposed with multi-layer recurrent neural networks. These time prediction methods rely on the black-box models, which prove to be more accurate, but fail to provide feedback to users.

In fact, the key requirement for users to adopt predictive techniques using deep neural networks are that they must have confidence in the predictive techniques or at least believe that the given predictions are un-

Manuscript Received May 15, 2021; Accepted Dec. 1, 2021. This work was supported by the National Natural Science Foundation of China (U1931207, 61702306), Sci. & Tech. Development Fund of Shandong Province of China (ZR2017BF015, ZR2017MF027), the Humanities and Social Science Research Project of the Ministry of Education (18YJAZH017), Shandong Chongqing Science and Technology Cooperation Project (cstc2020jscx-lyjsAX0008), Sci. & Tech. Development Fund of Qingdao (21-1-5-zlyj-1-zc), the Shandong Postgraduate Education Quality Improvement Plan (SDYJG19075), Shandong Education Teaching Research Key Project (2021JXZ010), National Statistical Science Research Project (2021LY053), the Taishan Scholar Program of Shandong Province, SDUST Research Fund (2015TDJH102, 2019KJN024), and National Statistical Science Research Project in 2019 (2019LY49).

 DOI:10.23919/cje.2021.00.170

biased. A useful and understandable explanation for remaining time prediction becomes more challenging. In order to explain how the remaining time prediction techniques are performed and use these explanations to understand why the prediction is accurate, a method of explainable remaining time prediction of business processes based on reachability graph is proposed in this paper, which includes the following two steps.

In the first step, on the one hand, a Petri net and the corresponding reachability graph are found from the event log, and the transition occurrence vector can be obtained from the reachability graph. On the other hand, prefixes, the corresponding suffixes and remaining time labels are received from the event log. Then, prefixes and the corresponding suffixes of traces are clustered into different transition partitions (corresponding to the sub-processes of the process model) by the activities of the last events of these prefixes. Moreover, the bidirectional recurrent neural networks with attention are used to encode the prefixes based on prefixes and the corresponding suffixes of the event log for each transition partition. Finally, the deep transfer learning between different transition partitions based on the transition occurrence vector is operated to evaluate remaining time prediction more accurately. In the second step, the prefix and corresponding suffix of the running traces (process instances) are generated, which can verify the predictive models and obtain the evaluation values. Then, the evaluation values are added to the sub-processes of a Petri net corresponding to different transition partitions, which realizes the visualization of the prediction models.

The main contributions of this paper: First, a more accurate prediction method for business process remaining time based on reachability graphs and recurrent neural networks (gate recurrent units, GRUs) with attention is proposed by combining process models and deep neural networks. Second, a visualization of the prediction models is given in the form of process models (e.g., Petri nets), which improves the interpretability of why the remaining time prediction is accurate.

II. Related Work

We first provide brief introduction remaining time prediction of business processes and then focus on explainable predictive process monitoring.

1. Business process remaining time prediction

Business process remaining time prediction using machine learning (especially deep learning) has become a research hotspot of process prediction. In [13], a method to predict the remaining time of a running case is presented. First, the future paths of cases are predicted based on an annotated transition system with fuzzy support vector machine probabilities by the event logs. The remaining time of running traces is then predicted by adding the durations of future activities, which are each estimated by a support vector regressor. In [14], a method based on Bayesian Neural Networks to predict the remaining time in a instance is proposed. Specifically, the historical process instance is used to generate feature vectors that integrate the dependencies within the process instances and the dependencies between the process instances, and the prediction model of the remaining execution time is trained based on these feature vectors. In [15], a deep neural network by entity embedding is used, which combines a deep neural network and entity embedding to improve the predictive performance of the remaining time. In [16], a time-oriented interactive process miner is proposed for the remaining time prediction, which predicts the remaining time of each trace in a business workflow. Moreover, an adversarial framework for predicting the next timestamp via the generative adversarial net is presented is presented in [17].

2. Explainable predictive business process monitoring

Several approaches to explain predictive business process monitoring have been proposed, which has received significant attention in recent years.

The essential problem of explaining capabilities for predictive business process monitoring is tackled in [18]. Therefore, the reasons are reported when predicting generic KPIs. In [19], a novel approach to explain why a prediction model gives wrong predictions is proposed. It leverages post-hoc explainers and different encodings to identify features that induce a predictor to make mistakes, and eventually improve its accuracy. In [20], a visualization technique is proposed that uses gated graph neural networks to make decisions easier to interpret. In [21], a local post hoc interpretation method is proposed for making the adopted deep learning methods interpretable.

In summary, the existing predictive business process monitoring rarely explains the remaining time prediction method of business processes. In this paper, explainable remaining time prediction of business processes is proposed.

III. Remaining Time Prediction

The remaining time of the trace represents the difference between the completion time of the trace and the current time. Given a trace $\sigma =< e_1, e_2, \ldots, e_m >$ $(m \geq 1)$, the remaining time of the trace is defined as $\mathrm{rem}(\sigma, i) = \mathrm{time}(e_{|\sigma|}) - \mathrm{time}(e_i)$, where $\mathrm{time}(e_{|\sigma|})$ indicates the completion time of the trace and $\mathrm{time}(e_i)$ indic-

ates current time for the execution of e_i of the trace.

The remaining time prediction task under the machine learning framework is composed of training and application. The training is to construct prediction models by the event log. Firstly, the training set is derived from the event log. Moreover, the training samples can be denoted as $\mathrm{Da} = \{(\mathrm{VR}(\sigma), \mathrm{rem}(\sigma, i)) \,|\sigma \in L, 1 \leq i \leq |\sigma|\}$, where $\mathrm{VR}(\sigma)$ means the feature vector of a trace and $\mathrm{rem}(\sigma, i)$ represents the real time of a trace. Secondly, the mapping function $f : \varepsilon^* \rightarrow R^+$ on the training set is trained for the optimization goal. Moreover, the optimization goal is the error of the training set, supplemented by the regularization term, namely $f^* = \arg\max_f \sum_{(\sigma_i, t_i) \in \mathrm{Da}} (f(\sigma_i) - t_i)^2 + \Omega(f)$. For application, the models are applied to predict the time of the running traces. For a running trace σ_i, which i denotes the number of events in which σ_i have occurred, the mapping is utilized to estimate its total execution time of a trace, i.e. $\mathrm{rem}(\sigma, i) = f(\sigma_i)$.

IV. Explainable Method

The proposed method is described. First, an overview of the prediction method framework is presented, and then the key parts of the proposed method are highlighted.

Fig.1 shows our framework for explainable business process remaining time prediction using reachability graph. The framework mainly includes two steps: prediction models and visualization of prediction models.

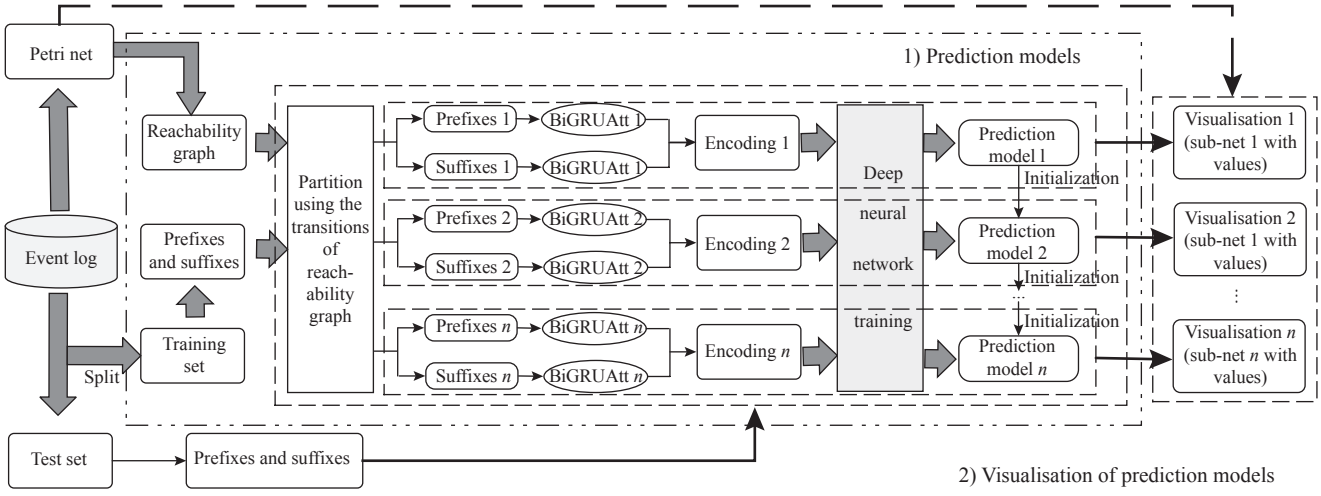


Fig. 1. A framework of our method.

1) Prediction models. On the one hand, a Petri net and its initial marking are mined from the event log. Further, the reachability graph is gained based on a Petri net and the corresponding markings, and the transition occurrence vector is obtained by the reachability graph. On the other hand, the training set and the test set are completely separated from the log. Regarding the training set, prefixes and the corresponding suffixes of the training set are generated from it. Then, prefixes and suffixes are divided into different transition partitions using the activities of the last events of prefixes by transition occurrence vector of the reachability graph. Next, for each transition partition, the prefixes are encoded based on the bidirectional GRUs with attention to get representations of the prefixes. And the suffixes are encoded based on the bidirectional GRUs with attention to get representations of the suffixes. The representation of the prefixes and the representation of the suffixes are concatenated as the encoding of the prefixes. Finally, deep transfer learning is performed on different deep neural networks according to the transition execution sequence of the reachability graph, and the prediction models of different transition partitions are obtained.

2) Visualization of prediction models. Regarding the test set, prefixes and the corresponding suffixes are generated. With prefixes and the corresponding suffixes of the test set, we can verify the final effect of the model and get the evaluation values. Then, the evaluation values are added to the sub-processes (sub-nets) of a Petri net corresponding to different transition partitions to realize the visualization of the prediction models.

1. Partition using the transitions of reachability graph

The basic concepts in predictive process monitoring and process models are discussed. Next, partition of prefixes and the corresponding suffixes using reachability graph is provided in this section.

1) Event log

Definition 1 (Event) [22] An event means an in-

stance of the execution of an event for a business process, $e=(a,c,\text{time}_s,\text{time}_e,(d_1,v_1),(d_2,v_2),\ldots,(d_n,v_n))$ $(n\geq 0)$, where a is the activity (transition) of event e, and c is identifier (ID) of the case to which e belongs. time_s and time_e are the timestamp of e, indicating the start and completion time of the event, respectively. (d_i,v_i) $(1\leq i\leq n)$ describes attributes of the event, where d_i $(1\leq i\leq n)$ denotes the attribute of the event, and v_i $(1\leq i\leq n)$ denotes the value corresponding to each attribute, respectively.

Definition 2 (Trace) [22] A trace (case or process instance) $\sigma=<e_1,e_2,\ldots,e_m>$ $(m\geq 1)$ refers to a non-empty sequence of events generated. And the timestamp is non-decreasing, i.e., $e_i.\text{time}_e\leq e_{i+1}.\text{time}_e$ $(1\leq i<i+1\leq m)$.

Definition 3 (Prefix, suffix) [23] $\sigma=<e_1,e_2,\ldots,e_m>$ $(m\geq 1)$ is given, $k\in\{1,2,\ldots,m\}$ be a positive integer. The event prefix hd^k and suffix tl^k of length k can be denoted as: $\text{hd}^k=<e_1,e_2,\ldots,e_k>$ and $\text{tl}^k=<e_{k+1},e_{k+2},\ldots,e_m>$. The activity prefix and suffix can be denoted as: $\pi_A(\text{hd}^k)=<\pi_A(e_1),\pi_A(e_2),\ldots,\pi_A(e_k)>$ and $\pi_A(\text{tl}^k)=<\pi_A(e_{k+1}),\pi_A(e_{k+2}),\ldots,\pi_A(e_m)>$.

A trace $\sigma=<a,b,c>$ is given, $<a>$ and $<a,b>$ are the prefixes of σ, and $<b,c>$ and $<c>$ corresponding to each prefix in prefixes of σ, respectively.

2) Process model

A process model provides a visual representation of the event log, and Petri net is utilized as the formal representation of the process model in this paper.

A (Petri) net [24], [25] $N=(P,T,F,W)$, such that P denotes a finite set of places, T denotes a finite set of transitions, $P\cap T=\varnothing$, $P\cup T\neq\varnothing$, $F\subseteq(P\times T)\cup(T\times P)$ denotes a set of flow relation, and W denotes a weight function, i.e., $W:((P\times T)\cup(T\times P))\to N^+$ denotes the weights of the arcs. A marking of the net N denotes the mapping from P to N^+, defining the number of tokens in each place of the net N. A system is a tuple $S=(N,M_0)$, where N is a net and $M_0\in P\to N^+$ is an initial marking (state). The post-set $n^\bullet$, and pre-set ${}^\bullet n$ of a node $n\in(P\cup T)$ are denoted as $n^\bullet=\{n'\in P\cup T\,|W(n,n')>0\}$ and ${}^\bullet n=\{n'\in P\cup T|W(n',n)>0\}$. In this paper, $W(n,n')=W(n',n)=1$.

Consider a system $S=(N,M_0)$ with $N=(P,T,F,W)$. A transition t is enabled at M_0 if for each p in ${}^\bullet t$, $M_0(p)\geq W(p,t)$, in which t is fireable from M_0. The firing of t from M_0 leads to the marking M, and is denoted by $M_0\,[t\rangle\,M$. A finite (firing) trace $\sigma=<e_1,e_2,\ldots,e_m>$ $(m\geq 1)$, its transition sequence, denoted by $\sigma=<t_1,t_1,\ldots,t_m>$ $(m\geq 1)$. In this paper, we refer to the longest sequence with all the different transitions for transition sequences as transition occurrence vectors.

A trace σ is enabled in S if the successive states acquired, denoted by $M_0\,[t\rangle\,M_1\ldots M_{m-1}\,[t\rangle\,M_m$, satisfy $M_k\,[t\rangle\,M_{k+1}$ $(k\in\{1,2,\ldots,m\})$, in which M_m is reachable from M_0, i.e., $M_0\,[t\rangle\,M_m$. The markings reachable from M_0 is expressed by $R(M_0)$ or $[M_0\rangle$. A reachability graph of S, i.e., $\text{RG}(S)$, denotes the directed graph (V,A,ι), where V denotes the vertices labeled with the markings $[M_0\rangle$, A denotes the arcs labeled with transitions of T such that the arc $M\xrightarrow{t}M'$ if and only if $M\,[t\rangle\,M'$ and $M\in[M_0\rangle$, and ι is the root, labeled with M_0. Since the prediction uncertainty of a trace with a length of 1 or 2 is relatively high and it affects the accuracy of the prediction, the prefix with a length greater than or equal to 3 is retained in this paper.

3) Partition of prefixes and the corresponding suffixes using reachability graph

First, a Petri net and the corresponding marking are mined from the event log by process mining algorithms. In this paper, inductive miner [26] is used. Then, the reachability graph is obtained by Petri net and the corresponding markings. In this partition phase, only the control flow perspective is considered.

The execution direction of the transition occurrence vector is consistent with the execution order of the transition of the reachability graph. In addition, regarding the different transitions of the concurrent structure in process model, the order in the transition occurrence vector is determined according to the order in which these transitions are recorded in the event log, that is, the transitions that are recorded preferentially are placed in front of the transitions that are recorded later.

According to the complete transition occurrence vector of the reachability graph of a Petri net, prefixes and the corresponding suffixes from the event log can be divided into different partitions based on the transition of the last event of the prefixes from the event log. Furthermore, each partition corresponds to a sub-process of the process model. In other words, the prefixes of each partition have similar structural information. Such partitioning of prefixes can improve the targeted training of recurrent neural networks to encode prefixes. The partitioning algorithm of prefixes and suffixes from the event log is shown in Algorithm 1.

Algorithm 1 Partition of prefixes and suffixes

Require: event log, EL.

Ensure: Petri net, PN; partitioned class, PC.

1: $\text{Prefix},\text{Suffix},\text{PC}=\varnothing$;

2: $\text{PN}+\text{markting}\xleftarrow{\text{generate}}\text{Inductive\_Miner(EL)}$;

3: $\text{reachability graph}\xleftarrow{\text{generate}}\text{PN}+\text{markting}$;

4: $\text{TOV}=<\sigma_1,\sigma_2,\ldots,\sigma_{N^+}>(N^+=1,2,3,\ldots)$ $\xleftarrow{\text{obtain}}\text{Reachability Graph(RG)}$;

5: For each $\sigma\in\text{EL}$

6: $\text{Prefix}\xleftarrow{\text{genarate}}\sigma$;

7: Suffix $\overset{\text{genarate}}{\longleftarrow}$ σ;
8: EndFor
9: For each $t_k \in$ TOV
10: $P_k \leftarrow t_k$, $k = 1, 2, \ldots, N^+$;
11: EndFor
12: $P_k = \varnothing$, $k = 1, 2, \ldots, N^+$;
13: For each $\sigma_p \in$ Prefix
14: If $\sigma_p[-1] == t_k$
15: $P_k \leftarrow (\sigma_p \in \text{Prefix}, \text{label}), (\sigma_s \in \text{Suffix}, \text{label})$;
16: EndIf
17: EndFor
18: PC $\leftarrow \{P_k | k = 1, 2, \ldots, N^+\}$;
19: Return PN, PC.

In Algorithm 1, Line 1 initializes the lists of Prefix, Suffix and partitioned class. Line 2 mines a Petri net and the corresponding markings from the event log by process mining algorithm. Line 3 generates the reachability graph by a Petri net and the corresponding markings. Line 4 obtains the complete transition occurrence vector from the reachability graph. Lines 5 to 8 generate prefixes and the corresponding suffixes. Lines 9 to 11 name the partitioned classes based on the transitions from the complete transition occurrence vector. Line 12 initializes each partition of the partitioned class. Lines 13 to 17 realize the partitions of all the prefixes and suffixes from the event log. Furthermore, Lines 14 to 16 add (prefixes, the remaining time) and (the corresponding suffixes, the remaining time) to each transition partition. Different transition partitions are sorted based on the sequence of transitions in Line 18. Finally, the partitioned class list and a Petri net are returned in Line 19.

For ease of comprehensibility, a simple example is given to illustrate the proposed prediction method. Given a simple log in Table 1, which contains four traces, and each trace is composed activities of events and timestamps. First, the process model (shown in Fig.2) and the corresponding reachability graph (shown in Fig.3) are discovered from the simple log. Then, we can get the transition occurrence vector [a, b, c, d, e, f] from the reachability graph. Furthermore, transitions b and c are in a concurrent relationship. Since transition b appears earlier than transition c in the log, transition b is before transition c of the transition occurrence vector.

Table 1. A log

Case ID	Traces
1	$< a^0, b^2, c^3, d^5, f^7 >$
2	$< a^3, c^5, b^6, d^8, f^9 >$
3	$< a^1, b^4, c^5, d^7, e^8, d^{10}, f^{11} >$
3	$< a^1, b^4, c^5, d^7, e^8, d^{10}, e^{11}, d^{13}, f^{14} >$

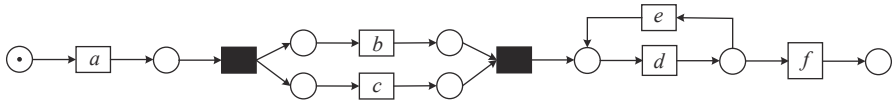


Fig. 2. A Petri net.

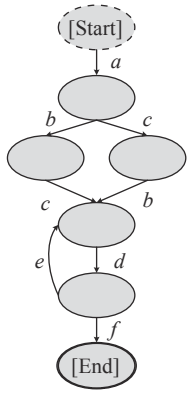


Fig. 3. The reachability graph of a Petri net.

At the same time, the prefix and the corresponding suffix of each trace in the simple log are generated, and the prefixes with a length greater than or equal to three, and the corresponding suffixes are partitioned based on the complete transition occurrence vector [b, c, d, e, f]. The partition rule is to analyze whether the activity of the last event of the prefix is the same as the transition in the transition occurrence vector. If so, the prefixes, the corresponding suffixes and the remaining time label are stored in the transition partition. The partitions of the simple log are shown in Fig.4. It is not difficult to find that each transition partition corresponds to the sub-process of the process model.

2. Bidirectional recurrent neural networks with attention on prefixes and suffixes

Bidirectional recurrent neural network with attention and the prediction models on prefixes and suffixes are described.

1) Bidirectional recurrent neural network with attention

Recurrent neural network (RNN) is a deep neural network model for sequence processing. The basic structure of the recurrent neural network is shown in Fig.5.

The most representative RNN neurons are LSTM [28] and GRU [29], which effectively solve the problems of gradient disappearance and short-term memory in traditional RNN. In this paper, we chose GRU as the implementation of RNN neurons.

The formulas for GRU neurons are: $\boldsymbol{h_t} = \text{GRU}(\boldsymbol{x_t},$

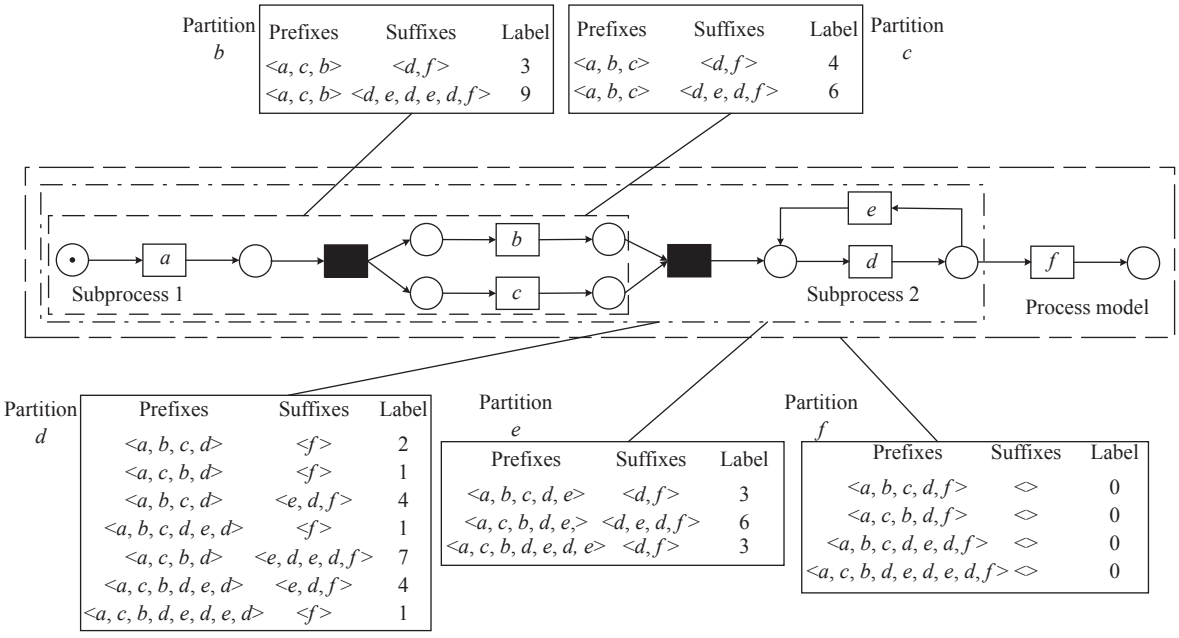


Fig. 4. Partitions of a simple log.

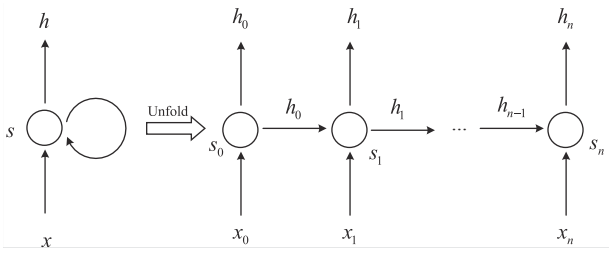


Fig. 5. Basic structure of recurrent neural network [27].

$\boldsymbol{h_{t-1}}$), $\boldsymbol{r_t}=\sigma(\boldsymbol{W}^{\text{reset}}\cdot[\boldsymbol{h_{t-1}},\boldsymbol{x_t}]+\boldsymbol{b}^{\text{reset}})$, $\boldsymbol{z_t}=\sigma(\boldsymbol{W}^{\text{update}}\cdot[\boldsymbol{h_{t-1}},\boldsymbol{x_t}]+\boldsymbol{b}^{\text{update}})$, $\widetilde{\boldsymbol{h_t}}=\tanh(\boldsymbol{W^h}[\boldsymbol{r_t}\odot\boldsymbol{h_{t-1}},\boldsymbol{x_t}]+\boldsymbol{b^h})$, $\boldsymbol{h_t}=(1-\boldsymbol{z_t})\odot\boldsymbol{h_{t-1}}+\boldsymbol{z_t}\odot\widetilde{\boldsymbol{h_t}}$, where $[,]$ means vector connection, $\cdot$ means matrix/vector multiplication, $\odot$ is an element-wise multiplication, σ is sigmoid activation function, and $\tanh(\cdot)$ is hyperbolic tangent function. The parameters of the GRU are the transformation matrix $\boldsymbol{W}^{\text{reset}}$, $\boldsymbol{W}^{\text{update}}$ and $\boldsymbol{W^h}$, and the corresponding bias $\boldsymbol{b}^{\text{reset}}$, $\boldsymbol{b}^{\text{update}}$ and $\boldsymbol{b^h}$.

Bidirectional recurrent neural network [30] is proposed to train models. It considers the correlation between events of traces. Let the output hidden vectors obtained by the forward RNN and the backward RNN be $\overrightarrow{\boldsymbol{h_t}}$ and $\overleftarrow{\boldsymbol{h_t}}$, respectively, and concatenate them to obtain the context encoding $\boldsymbol{h_t}=[\overrightarrow{\boldsymbol{h_t}},\overleftarrow{\boldsymbol{h_t}}]$ for each moment.

The encoding of the trace is constructed by the context encoding at each moment of the trace, and the encoding of the trace is as follows: $\boldsymbol{v}=\sum_{k=1}^{n}\boldsymbol{\alpha_t}\boldsymbol{h_t}$, where $\boldsymbol{\alpha_t}$ is the weight of the context encoding at time t. In this paper, a multi-layer perceptron (called attention) is used to calculate the context weight $\boldsymbol{\alpha_t}=\text{softmax}(\boldsymbol{g}^{\text{attentionT}}\cdot\tanh(\boldsymbol{W}^{\text{attention}}\cdot\boldsymbol{h_t}+\boldsymbol{b}^{\text{attention}}))$, where $\boldsymbol{g}^{\text{attentionT}}$, $\boldsymbol{W}^{\text{attention}}$ and $\boldsymbol{b}^{\text{attention}}$ are the parameters of the attention mechanism.

2) Prediction models on prefixes and suffixes

For each transition partition of the partitioned class, different prediction models are designed to predict the remaining time of business processes. The network structure of the prediction model for each transition partition is shown in Fig.6. Specifically, looking at it from the bottom up, first of all, regarding prefix encoding, prefixes are used as input, and bidirectional GRUs are used for training to obtain the context encodings of prefixes for each moment. The vector representation v of prefixes can be obtained by the attention mechanism. Similarly, regarding suffix encoding, the representation of the suffixes is used as input, and the vector representation v' of suffixes is obtained by using bidirectional GRUs with attention. Then, the vector representation v of prefixes and the vector representation v' of suffixes are concatenated as the vector representation $\overline{v}$ of the prefixes. Finally, a multi-layer perceptron is used to construct prediction model. The specific calculation formula is $\text{Remain\_time}=\boldsymbol{g}^{\text{T}}\cdot\text{ReLU}(\boldsymbol{W}\cdot\overline{v}+\boldsymbol{b})$, where $\boldsymbol{g}^{\text{T}}$, $\boldsymbol{W}$ and $\boldsymbol{b}$ are the model parameters of the multilayer perceptron. In this paper, the linear rectification function (rectified linear unit, ReLU) is the activation function.

For each transition partition, the encoding method of concatenating the prefix encoding and the corresponding suffix encoding as the prefix can fully express the structural information of the prefix, thereby improving the accuracy of prediction. The encodings on prefixes and suffixes are shown in Algorithm 2.

In Algorithm 2, Line 1 initializes vector representa-

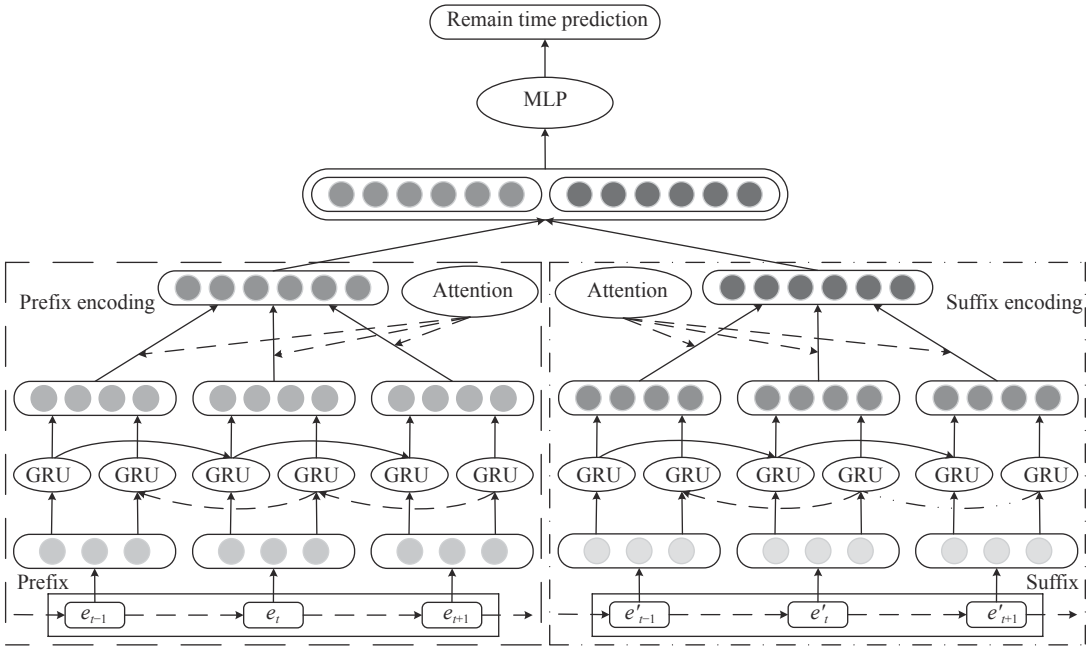


Fig. 6. Prediction model for each transition partition.

tion of prefixes. Lines 2 to 9 encode prefixes and the corresponding suffixes of each transition partition in the partitioned class using the bidirectional recurrent neural networks with attention, respectively. Lines 3 to 5 encode prefixes by the bidirectional GRUs with attention, and Lines 6 to 8 encode the corresponding suffixes by the bidirectional GRUs with attention. Line 10 concatenates vector representation of prefixes and corresponding suffixes as prefixes encoding to train the prediction models, and vector representation of prefixes are returned in Line 11.

Algorithm 2 The encodings on prefixes and suffixes

Require: partitioned class, PC.
Ensure: vector representation of prefixes, VRP.
1: VRP $= \varnothing$;
2: For each $P_k \in$ PC $(k = 1, 2, \ldots, N^+)$
3: For each $\sigma_{\text{prefix}} \in P_k$
4: $\text{VRP}_{\text{prefix}} \leftarrow \text{Bi\_GRU\_Att}_i\ (\sigma_{\text{prefix}})$;
5: EndFor
6: For each $\sigma_{\text{suffix}} \in P_k$
7: $\text{VRP}_{\text{suffix}} \leftarrow \text{Bi\_GRU\_Att}_i\ (\sigma_{\text{suffix}})$;
8: EndFor
9: EndFor
10: VRP $\leftarrow$ concatenate $(\text{VRP}_{\text{prefix}}, \text{VRP}_{\text{suffix}})$;
11: Return VRP.

In order to verify that our encoding on prefixes and suffixes can more accurately predict the remaining time and better explain the reason for the accuracy, a comparison experiment with different encodings, including Prefix encoding, Suffix encoding and our method Prefix_Suffix encoding_Transfer, was performed on the publicly available event logs. Moreover, Prefix encoding only considers the encoding of the prefix as the encoding method of the prefix. Suffix encoding only considers the suffix encoding corresponding to the prefix as the encoding method of the prefix. Prefix_Suffix encoding_Transfer means that the vector representation of prefix and the corresponding suffix are concatenated as the encoding method of prefix.

3. Prediction models

The prefixes and the corresponding suffixes of the training set are divided according to the complete transition occurrence vector of the reachability graph, and deep transfer learning is performed on each transition partition in the order of occurrence of the complete transition occurrence vector. The proposed method can obtain better predictive performance because each transition partition corresponds to the sub-processes of the process model, and the transfer learning between the sub-processes is also consistent with the forward execution of the process model. Prediction model training is shown in Algorithm 3.

Algorithm 3 Prediction model training

Require: trace prefixes, σ; partitioned class, PC; vector representation of prefixes, VRP.
Ensure: prediction models, PM.
1: PC $\leftarrow$ PC(σ);
2: VRP $\leftarrow$ VRP(σ);
3: For each PC.index
4: If $\sigma[-1]$ == PC.index

5: $\sigma \in P_i$;
6: $\text{Model}_i \leftarrow P_i$;
7: EndIf
8: $\text{Model}_{i+1} \xleftarrow{\text{transfer}} \text{Model}_i$;
9: EndFor
10: PM $\leftarrow \{\text{Model}_i | i \in \text{PC.index}\}$;
11: Return PM.

In Algorithm 3, Line 1 partitions the prefixes of the training set. Line 2 encodes the prefixes. Lines 3 to 9 train the prediction models. Line 4 determines whether the activity of the last event of the prefix σ is the same as PC.index. If yes, Line 5 queries its corresponding transition partition. A prediction model is trained in Line 6 using the deep transfer learning for each transition partition. Line 8 describes the deep transfer learning that the parameters of Model_{i+1} are initialized with those of Model_i. Finally, the prediction models are obtained in Line 10, and the Prediction models is returned in Line 11.

4. Visualization of prediction models

Applying the running traces to the prediction models, and the evaluation values can be obtained. Then, add the evaluation values to the sub-processes corresponding to different partitions. Finally, the visualization of the prediction models in the form of sub-processes of a Petri net. Remain time prediction of running traces with visualization (RTPV) is shown in Algorithm 4.

Algorithm 4 RTPV

Require: running traces, σ_r; Petri net, PN; partitioned class, PC; vector representation of prefixes, VRP; prediction models, PM.
Ensure: visualization of prediction models, VPM.
1: PC $\leftarrow$ PC(σ_r);
2: VRP $\leftarrow$ VRP(σ_r);
3: For each PC.index
4: If $\sigma_r[-1] ==$ PC.index
5: $\sigma_r \in P_i$;
6: $\text{Model}_i \in \text{PM} \leftarrow P_i$;
7: EndIf
8: List $\leftarrow$ values;
9: EndFor
10: For each value in List
11: Sub-process-values $\xleftarrow{+}$each-values-of-List;
12: EndFor
13: VPM $\leftarrow$ Sub-process-values;
14: Return VPM.

In Algorithm 4, Line 1 partitions the running traces. Line 2 encodes the running traces. Lines 3 to 9 validate the prediction model for each transition partition and obtain the evaluation values. Then, the evaluation values of the prediction model for each transition partition are added to the sub-process of a Petri net in Lines 10 to 12. Line 13 visualizes the prediction model for each transition partition using the sub-process with the evaluation values of a Petri net. Finally, visualization of prediction models is returned in Line 14.

For example, the prefixes and suffixes in each transition partition can be separately encoded by the bidirectional recurrent neural networks with attention. And the prefix encoding and suffix encoding are concatenated together as the encoding of prefix to train the prediction model. Then, the transfer learning is performed in accordance with the execution order of the complete transition occurrence vector [b, c, d, e, f]. Specifically, the model parameters of the trained prediction model b are used as the initialization parameters of the prediction model c, and the model parameters of the trained prediction model c are used as the initialization parameters of the prediction model d, and the deep transfer learning continues until the training of the last prediction model f is completed (shown in Fig.7).

Given the running prefixes, we can obtain the evaluation values by applying the running prefixes to the prediction models. Further, the evaluation values (for example, MAE and MSE) of each prediction model are added to the sub-process corresponding to its transition partition, and the visualization of all the prediction models is shown in Fig.7.

V. Evaluation

We will conduct an experimental evaluation of the proposed method.

1. Event logs

We performed experiments by the publicly available event logs to validate the effectiveness of our proposed method. The Business Process Intelligence Challenge 2017_O [31] (BPIC_2017_O_L) subprocess and Business Process Intelligence Challenge 2017_W [31] (BPIC_2017_W_P) subprocess provided the event logs from a German financial institution. Moreover, the "O" indicates state of the offer changes, and the "W" indicates state of the work item for the loan application. Helpdesk [10] (Helpdesk_P) is derived from a real-life event log. It comes from an event of a ticket management process at the helpdesk of an Italian software company. These event logs can be downloaded at *http://www.processmining.org/logs/start*.

The infrequent behaviors of event logs are cleaned. The cleaned event logs:

a) BPIC_2017_O_L,
b) BPIC_2017_W_P, and
c) Helpdesk_P,

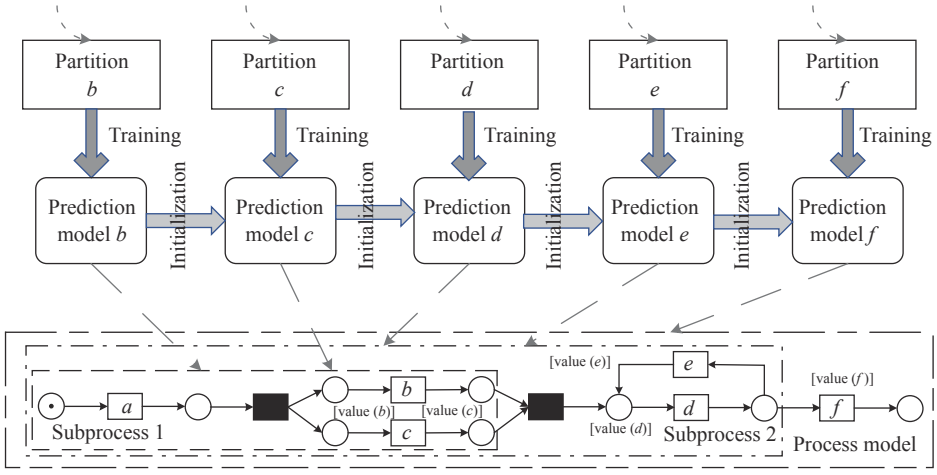


Fig. 7. Explainable process prediction by a example.

are used in our experiments. Table 2 presents statistics of the publicly available event logs used of experiments. Note that Nt represents number of traces, Ne denotes number of events, Na denotes number of activities, Minlt denotes minimum length of traces, and Maxlt denotes maximum length of traces. Moreover, BPIC_2017_O_L has loop structures and sequence structures. BPIC_2017_W_P and Helpdesk_P have parallel structures and sequence structures.

Table 2. Basic statistics of a), b), and c)

Event logs	Nt	Ne	Na	Minlt	Maxlt
a)	5842	21193	5	2	9
b)	5644	41516	6	6	17
c)	3695	13233	6	1	14

We split each event log into two parts. The first part (70% of traces from the event log) is used as a training set, and the remaining 30% of traces is used to evaluate the prediction medel.

2. Metrics

The evaluation metrics of our experiments include the mean absolute error (MAE) [32] and the mean squared error (MSE) [33], [34]. These calculation formulas are $\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|f(\sigma_i) - \mathrm{rem}(\sigma, i)|$ and $\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(f(\sigma_i) - \mathrm{rem}(\sigma, i))^2$. Moreover, n denotes the size of predicted samples, $f(\sigma_i)$ denotes the predicted value of the remaining time in the running trace σ_i, and $\mathrm{rem}(\sigma, i)$ denotes the real value of the remaining time.

3. Implementation

We implemented all the methods as a set of Python 3.7 scripts using the recurrent neural network library PyTorch 1.3.1. The experiments were performed on a single NVIDA RTX 2070 super GPU with 8 GB memory. Moreover, the discovery of Petri net models and their reachability graphs are implemented in ProM 6.9. Our experimental settings are as follows: 1) The learning rate is 0.01; 2) The number of iterations is 150; 3) The optimizer is Adam.

4. Results

We compare our method against several baselines shown in Table 3.

First, the methods using transition system (methods 1, 2, and 3 represent the set, sequence, and multiset abstraction, respectively) [35], data-aware transition

Table 3. MAE and RMSE values for a), b) and c)

Method	a)		b)		c)	
	MAE	MSE	MAE	MSE	MAE	MSE
1	17.266	898.278	10.915	351.925	14.292	334.048
2	6.708	118.552	11.110	351.780	15.631	355.492
3	17.213	867.884	10.868	348.009	14.497	335.806
4	6.715	118.676	6.032	133.894	6.351	114.418
5	6.708	118.552	5.962	132.317	6.297	109.717
6	6.715	118.676	6.028	134.044	6.389	114.274
7	0.273	4.679	6.002	117.165	2.866	51.676
8	0.275	4.775	6.065	116.745	3.097	62.247
9	0.231	3.273	6.326	139.736	2.673	44.680
10	0.347	8.843	6.328	145.053	2.612	44.855
11	1.521	45.249	5.119	105.158	2.820	54.901
12	**0.151**	**1.601**	**4.745**	**92.420**	**1.665**	**23.570**

system (methods 4, 5, and 6 represent the set, sequence, and multiset abstraction, respectively) [36], LSTM neural networks [37], GRU neural networks, transfer learning (methods 9 and 10) [12] based on LSTM (method 7), and GRU (method 8) are used as the baseline methods. The method of GRU neural networks is similar to the method of LSTM neural networks, and their difference is that the neuron is implemented in a different way.

Next, the transformer [38] encoder (method 11) with strong expressive ability is compared with GRU. Finally, 12 represents our method. Note that the experimental results of the existing remaining time prediction methods [7] show that the methods based on neural networks are superior to those based on machine learning. In this paper, neural network methods are selected as the baselines.

A parameter selection process for the methods based on deep neural networks is provided. We selected the same parameters of these methods for experiments, and the results are shown in Fig.8. In order to ensure the consistency of the parameters of the comparison method, we set the value of embedding size to 2 and the value of hidden size to 5. Our method has the lowest parameter sensitivity and is better than the competitive methods overall.

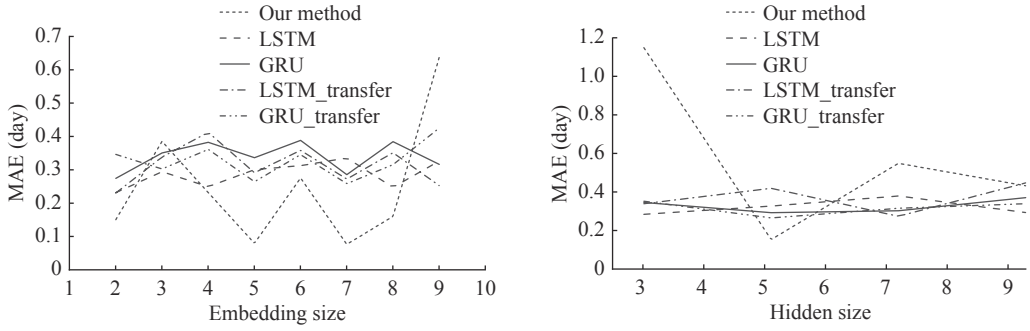


Fig. 8. Parameter selection process.

Regarding the comparison of model size, first of all, the model size of vanilla models of LSTM or GRU is not large. Then, the model size of transfer learning based on LSTM or GRU is a constant multiple of the model size of LSTM or GRU. The reason is that the number of transfers among LSTMs or GRUs is limited. Furthermore, the model size of our method is larger than that of these models, but it is only a constant multiple of the model size of these models.

The evaluation results of all the methods are shown in Table 3. The experimental results of all event logs containing loop and parallel structures are superior to the baseline methods, which shows that the proposed method has better performance for business process remaining time prediction. The reasons for the better performance of the proposed method are as follows: 1) The Prefix_Suffix encoding of traces can effectively express the structural and temporal features of traces; 2) The partition training of the prefixes can group the prefixes of similar structures to improve the accuracy of the prediction models; 3) According to the order of transition occurrence vector of the reachability graph, the proposed deep transfer learning among different prediction models takes into account the correlation between the sub-processes of the process model.

The experimental results of our method are analyzed in detail.

1) Petri nets and the corresponding reachability graphs

For each in a), b), and c), a Petri net and the corresponding reachability graph are generated by inductive miner. For a), the reachability graph RG($S(a)$) is shown in Fig.9 . Similarly, the reachability graph RG($S(b)$) for b) is shown in Fig.10. For c), the reachability graph RG($S(c)$) is shown in Fig.11.

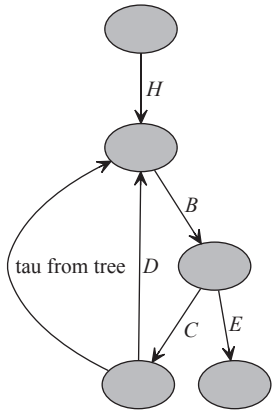


Fig. 9. The reachability graph RG($S(a)$) for a).

Statistics of the reachability graphs in Table 4. Number of vertices (Nv), Maximum in-degree of vertex (Maxiv), Maximum out-degree of vertex (Maxov), Minimum length of transition occurrence vector (Minl) and Maximum length of transition occurrence vector (Maxl)

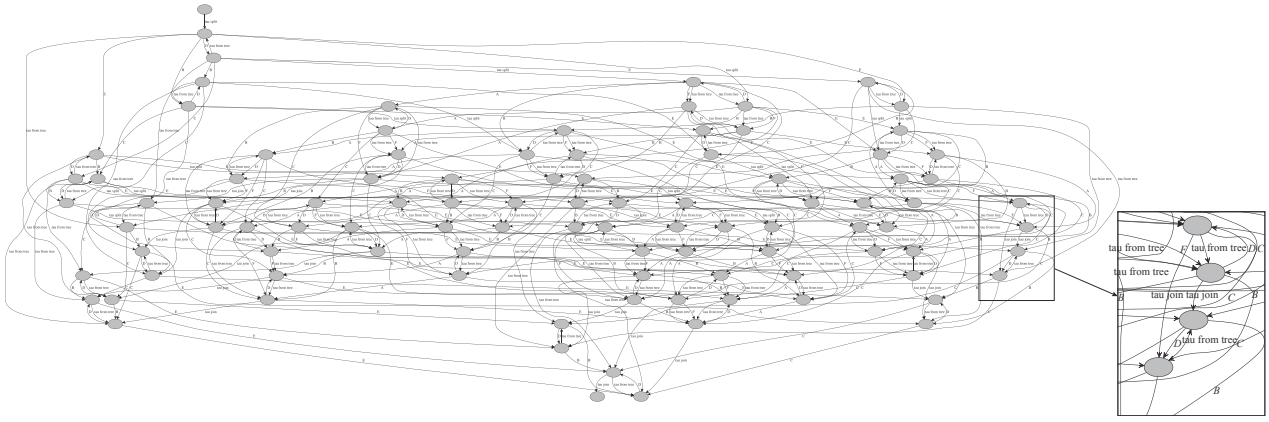


Fig. 10. A Petri net RG($S(b)$) for b).

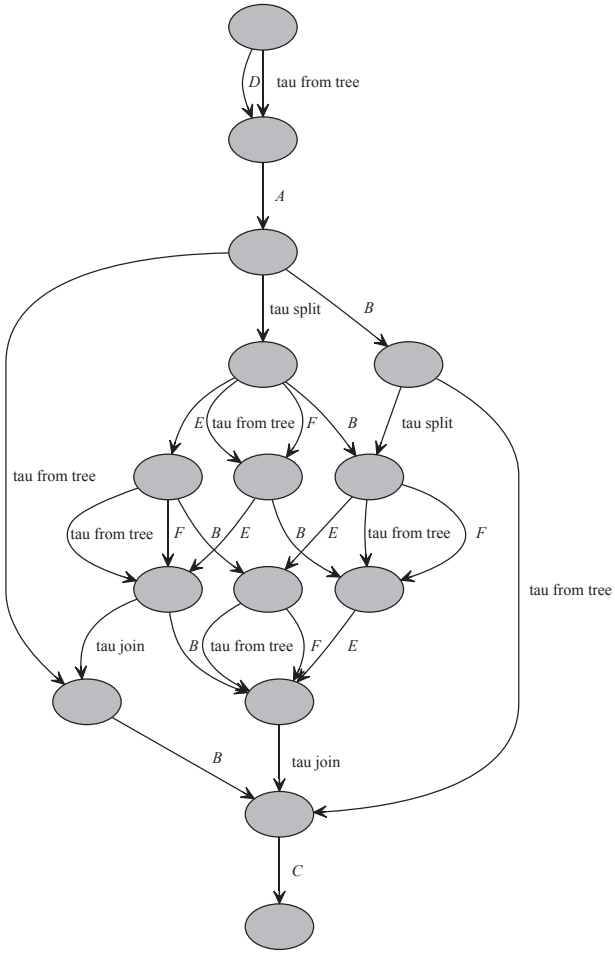


Fig. 11. The reachability graph RG($S(c)$) for c).

Table 4. Statistics of the reachability graphs

RG	Nv	Maxiv	Maxov	Minl	Maxl
RG($S(a)$)	5	2	2	3	5
RG($S(b)$)	98	5	5	6	6
RG($S(c)$)	15	3	3	4	6

reflect the structures in a Petri net, for example, concurrent or cyclic structures. In this paper, the complete transition occurrence vector refers to the transition occurrence vector with the largest length from the reachability graphs.

For a), the complete transition occurrence vector [*H, B, C, D, E*] is obtained from the reachability graph RG($S(a)$). For b), the corresponding complete transition occurrence vector [*F, A, B, C, D, E*] is also obtained from the reachability graph RG($S(b)$). For c), transition occurrence vector [*D, A, B, E, F, C*] is got from the reachability graph RG($S(c)$). Further, the order of recording transitions in the event log determines the order of transition occurrence vector for different transitions of concurrent structures.

In order to improve the accuracy of the prediction, the prefix with a length greater than or equal to three is trained in this paper. Therefore, the transition occurrence vector of the last event for all the prefixes may not be the complete transition occurrence vector of the reachability graph from initial marking. For a), the complete transition occurrence vector of the last event of all the prefixes is [*B, C, D, E*]. The complete transition occurrence vector of the last event of all the prefixes for b) is [*F, A, B, C, D, E*]. [*B, E, F, C*] is the complete transition occurrence vector of the last event of all the prefixes for c).

2) Evaluation of prediction models

Table 5 presents variants of our method Prefix_Suffix encoding_Transfer, which include Prefix encoding, Suffix encoding, and Prefix_Suffix encoding_NonTransfer. Prefix encoding refers to a transfer method that only relies on encoding of prefixes. Suffix encoding refers to a transfer method that only relies on encoding of suffixes. Prefix_Suffix encoding_NonTransfer refers to a non-transfer method that relies on encoding of prefixes and suffixes. Furtherore, our method Prefix_Suffix encoding_NonTransfer refers to a transfer method that relies on encoding of prefixes and suffixes.

Table 5. Variants of our method

Variants	Ep	Es	Tp	Dtl
Pe	√	×	√	√
Se	×	√	√	√
PSeT	√	√	√	√
PSeNT	√	√	√	×

MAE values of different variants containing Prefix encoding (Pe), Suffix encoding (Se), Prefix_Suffix encoding_NonTransfer (PSeNT) and our method Prefix_Suffix encoding_Transfer (PSeT) for a), b), and c) are shown in Fig.12. Note that Ep, Es, Tp and Dtl denote encoding of prefixes, encoding of suffixes, transition partitions and deep transfer learning, respectively. For all the event logs, MAE and MSE values of our method are lower than the MAE and MSE values of Prefix encoding, Suffix encoding, and Prefix_Suffix encoding_NonTransfer. The result shows that our method outperforms Prefix encoding and Suffix encoding for a), b), and c).

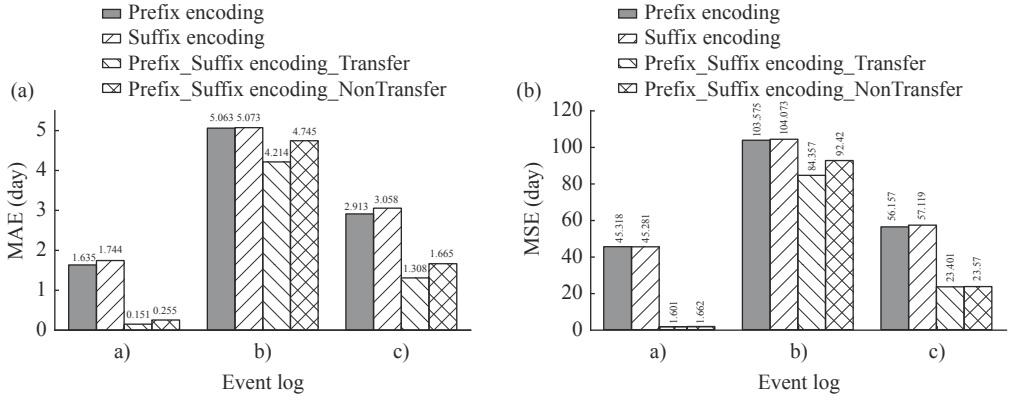


Fig. 12. Values of different variants for a), b) and c). (a) MAE; (b) MSE.

Furthermore, the MAE and MSE values of different variants for different transition partitions from the reachability graphs are presented for a), b), and c) in Fig.13.

For a), b), and c), except for the transition A partition of the event log b), the MAE and MSE values of our method are almost all lower than the MAE and MSE values of Prefix encoding, Suffix encoding, and Prefix_Suffix encoding_NonTransfer. In summary, the results also verify that our Prefix_Suffix encoding_Transfer can achieve better prediction performance to a certain extent.

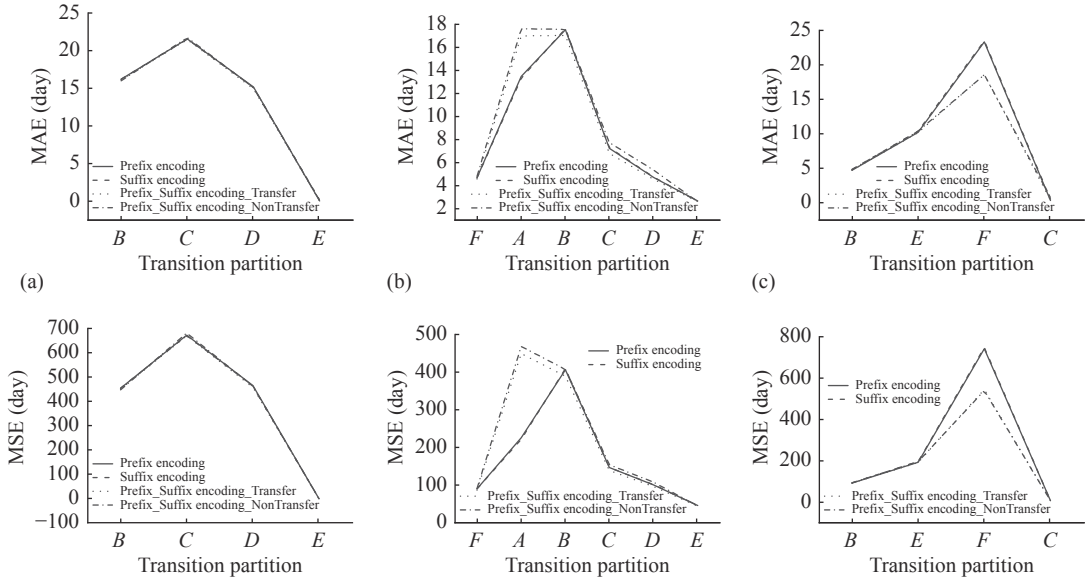


Fig. 13. Values on transition partitions for a), b), and c). (a) For a); (b) For b); and (c) For c).

The result of comparing our method Prefix_Suffix encoding_Transfer with Prefix encoding and Suffix encoding shows the superiority of our encoding on prefix and suffix. Moreover, by comparing our method Prefix_Suffix encoding_Transfer with Prefix_Suffix encoding_NonTransfer, the results show that the predic-

tion method using deep transfer learning outperforms the prediction methods of non-transfer learning. This is because the deep transfer learning considers the correlation between transitional partitions (sub-processes of the process model).

3) Visualization

MAE and MSE values of our method for different transition partitions are shown in Table 6. Furthermore, transition partitions of prefixes are shown in Fig.13, and - means none.

Table 6. MAE and MSE values of different transition partitions for a), b) and c)

a)		b)		c)	
MAE	MSE	MAE	MSE	MAE	MSE
16.128	448.252	4.625	86.890	4.786	92.326
21.474	665.423	16.991	448.584	10.247	191.787
14.950	455.272	17.045	388.333	18.596	542.565
0.098	0.016	6.849	139.125	0.492	7.685
–	–	4.597	96.929	–	–
–	–	2.672	44.496	–	–

For a), the MAE and MSE values [16.128; 448.252] of transition B partition are added to the sub-process B of a Petri net $S(a)$. The prediction model of transition B partition is explained by the sub-process B with values. Similarly, the MAE and MSE values [21.474; 665.423] of transition C partition are added to the sub-process C of a Petri net $S(a)$. The prediction model of transition C partition is explained by the sub-process C with values. The MAE and MSE values [14.950; 455.272] of transition D partition are added to the sub-process D of a Petri net $S(a)$. The prediction model of transition D partition is explained by the sub-process D with values. Finally, the MAE and MSE values [0.098; 0.016] of transition E partition represents the entire process model are added to a Petri net $S(a)$. The prediction model of transition E partition is explained by a Petri net $S(a)$ with values.

A visualization of prediction model for each transition partition in the shape of a process model are shown in Fig.14. Note that the visualization of the prediction model for each transition partition in the form of sub-process means that the activities of each sub-process are consistent with the activities of each transition partition.

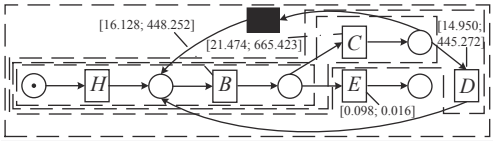


Fig. 14. A visualization of prediction models for a).

Similarly, for b), a visualization of prediction model for each transition partition in the shape of a process model are shown in Fig.15. For c), a visualization of prediction model for each transition partition in the form of a process model, as presented in Fig.16. Moreover, except for the outermost dashed box representing the entire process model, the remaining dashed boxes represent the sub-processes of transition partitions.

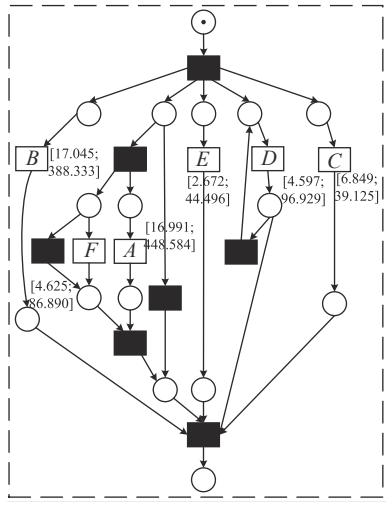


Fig. 15. A visualization of prediction models for b).

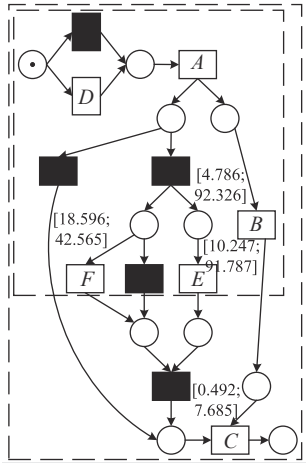


Fig. 16. A visualization of prediction models for c).

VI. Conclusions

A method of explainable business process remaining time prediction via structural process model is proposed in this paper. For training of prediction models, firstly, a Petri net and the reachability graph are mined from the event log, and the transition occurrence vectors can be obtained based on the reachability graph. Simultaneously, prefixes and suffixes are generated from the event log. Secondly, according to transition occurrence vectors of the reachability graph, prefixes and suffixes are clustered into different transition partitions using the activity of the last event for prefix. Furthermore, for each transition partition, the bidirectional re-

current neural network is used to encode the prefixes on prefixes and suffixes. Finally, the deep transfer learning between different transition partitions using the transition occurrence vectors is performed to predict remaining time. For the visualization of prediction models, the evaluation values are added to the sub-processes of a Petri net corresponding to different transition partitions which realize the visualization of the prediction models. In future, the prediction model not only depends on the control flow of traces, but also considers the context attributes of process instances improve the accuracy of predictions. Then, we plan to apply explainable techniques to other fields [39]–[41].

References

[1] C. Di Francescomarino, C. Ghidini, F. M. Maggi, *et al.*, "Predictive process monitoring methods: Which one suits me best?," in *Proceedings of the 16th International Conference on Business Process Management, Sydney*, Australia, pp.462–479, 2018.

[2] N. Harane and S. Rathi, " Comprehensive survey on deep learning approaches in predictive business process monitoring," in *Modern Approaches in Machine Learning and Cognitive Science: A Walkthrough*, V. K. Gunjan, *et al.*, Eds., Springer, Cham, pp.115–128, 2020.

[3] W. Van Der Aalst, *Process Mining: Data Science in Action*. Springer, Berlin, 2016.

[4] I. Teinemaa, M. Dumas, M. La Rosa, *et al.*, "Outcome-oriented predictive process monitoring: Review and benchmark," *ACM Transactions on Knowledge Discovery from Data*, vol.13, no.2, article no.17, 2019.

[5] S. Weinzierl, S. Zilker, J. Brunk, *et al.*, "An empirical comparison of deep-neural-network architectures for next activity prediction using context-enriched process event logs," *arXiv preprint*, arXiv: 2005.01194, 2020.

[6] J. Theis and H. Darabi, " Decay replay mining to predict next process events," *IEEE Access*, vol.7, pp.119787–119803, 2019.

[7] I. Verenich, M. Dumas, M. La Rosa, *et al.*, " Survey and cross-benchmark comparison of remaining time prediction methods in business process monitoring," *ACM Transactions on Intelligent Systems and Technology*, vol.10, no.4, article no.34, 2019.

[8] F. Taymouri and M. La Rosa, "Encoder-decoder generative adversarial nets for suffix generation and remaining time prediction of business process models," *arXiv preprint*, arXiv: 2007.16030, 2020.

[9] M. Camargo, M. Dumas, and O. González-Rojas, "Discovering generative models from event logs: Data-driven simulation vs deep learning," *PeerJ*, vol.7, article no.e577, 2021.

[10] A. Nguyen, S. Chatterjee, S. Weinzierl, *et al.*, " Time matters: time-aware LSTMs for predictive business process monitoring," in *Proceedings of 2020 International Workshops on Process Mining Workshops*, Padua, Italy, pp.112–123, 2021.

[11] M. Camargo, M. Dumas, and O. González-Rojas, "Learning accurate LSTM models of business processes," in *Proceedings of the 17th International Conference on Business Process Management*, Vienna, Austria, pp.286–302, 2019.

[12] T. Liu, W. J. Ni, Y. J. Sun, *et al.*, " Predicting remaining business time with deep transfer learning," *Data Analysis and Knowledge Discovery*, vol.4, no.S2, pp.134–142, 2020. (in Chinese)

[13] I. Firouzian, M. Zahedi, and H. Hassanpour, "Investigation of the effect of concept drift on data-aware remaining time prediction of business processes," *International Journal of Nonlinear Analysis and Applications*, vol.10, no.2, pp.153–166, 2019.

[14] G. Park and M. Song, "Predicting performances in business processes using deep neural networks," *Decision Support Systems*, vol.129, article no.113191, 2020.

[15] N. A. Wahid, T. N. Adi, H. Bae, *et al.*, " Predictive business process monitoring - Remaining time prediction using deep neural network with entity embedding," *Procedia Computer Science*, vol.161, pp.1080–1088, 2019.

[16] İ. Yürek, D. Birant, Ö. E. Yürek, *et al.*, "Time-oriented interactive process miner: A new approach for time prediction," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol.29, no.1, pp.122–137, 2021.

[17] F. Taymouri, M. La Rosa, S. Erfani, *et al.*, "Predictive business process monitoring via generative adversarial nets: The case of next event prediction," in *Proceedings of the 18th International Conference on Business Process Management,* Seville, Spain, pp.237–256, 2020.

[18] R. Galanti, B. Coma-Puig, M. De Leoni, *et al.*, "Explainable predictive process monitoring," in *Proceedings of 2020 2nd International Conference on Process Mining (ICPM)*, Padua, Italy, pp.1–8, 2020.

[19] W. Rizzi, C. Di Francescomarino, and F. M. Maggi, "Explainability in predictive process monitoring: when understanding helps improving," in *Proceedings of International Conference on Business Process Management,* Seville, Spain, pp.141–158, 2020.

[20] M. Harl, S. Weinzierl, M. Stierle, *et al.*, " Explainable predictive business process monitoring using gated graph neural networks," *Journal of Decision Systems*, vol.29, no.sup1, pp.312–327, 2020.

[21] N. Mehdiyev and P. Fettke, " Explainable artificial intelligence for process mining: A general overview and application of a novel local explanation approach for predictive process monitoring," in *Interpretable Artificial Intelligence: A Perspective of Granular Computing*, W. Pedrycz and S. M. Chen, Eds. Springer, Cham, pp.1–28, 2021.

[22] I. Verenich, "*Explainable predictive monitoring of temporal measures of business processes*," *Ph.D. Thesis*, Queensland University of Technology, Brisbane, 2018.

[23] E. Rama-Maneiro, J. C. Vidal, and M. Lama, "Deep learning for predictive business process monitoring: Review and benchmark," *IEEE Transactions on Services Computing*, vol.16, no.1, pp.739–756, 2023.

[24] T. Hujsa, B. Berthomieu, S. D. Zilio, *et al.*, "Checking marking reachability with the state equation in Petri net subclasses," *arXiv preprint*, arXiv: 2006.05600, 2020.

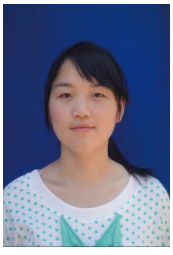
[25] T. Murata, " Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol.77, no.4, pp.541–580, 1989.

[26] S. J. J. Leemans, D. Fahland, and W. M. P. Van Der Aalst, "Discovering block-structured process models from event logs - a constructive approach," in *Proceedings of the 34th International Conf. on Applications and Theory of Petri Nets and Concurrency*, Milan, Italy, pp.311–329, 2013.

[27] Y. LeCun, Y. Bengio, and G. Hinton, " Deep learning," *Nature*, vol.521, no.7553, pp.436–444, 2015.

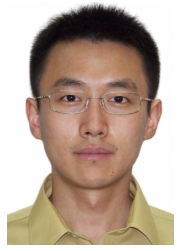
[28] S. Hochreiter and J. Schmidhuber, " Long short-term memory," *Neural Computation*, vol.9, no.8, pp.1735–1780,

1997.
[29] J. Chung, C. Gulcehre, K. H. Cho, *et al.*, Empirical evaluation of gated recurrent neural networks on sequence modeling, arXiv preprint arXiv: 1412.3555, 2014.
[30] M. Schuster and K. K. Paliwal, " Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol.45, no.11, pp.2673–2681, 1997.
[31] L. Lin, L. J. Wen, and J. M. Wang, "MM-pred: A deep predictive model for multi-attribute event sequence," *Proceedings of the 2019 SIAM International Conference on Data Mining*, Calgary, Canada, pp.118–126, 2019.
[32] N. Navarin, B. Vincenzi, M. Polato, *et al.*, " LSTM networks for data-aware remaining time prediction of business process instances," in *Proceedings of 2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, Honolulu, HI, USA, pp.1–7, 2017.
[33] 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 *Proceedings of OTM Confederated International Conferences " On the Move to Meaningful Internet Systems",* Monterrey, Mexico, pp.319–336, 2008.
[34] W. Ni, M. Yan, T. Liu, *et al.*, "Predicting remaining execution time of business process instances via auto-encoded transition system," *Intelligent Data Analysis*, vol.26, no.2, pp.543–562, 2022.
[35] W. M. P. Van Der Aalst, M. H. Schonenberg, and M. Song, "Time prediction based on process mining," *Information Systems*, vol.36, no.2, pp.450–475, 2011.
[36] M. Polato, A. Sperduti, A. Burattin, *et al.*, " Time and activity sequence prediction of business process instances," *Computing*, vol.100, no.9, pp.1005–1031, 2018.
[37] N. Tax, I. Verenich, M. La Rosa, *et al.*, " Predictive business process monitoring with LSTM neural networks," in *Proceedings of the 29th International Conference on Advanced Information Systems Engineering*, Essen, Germany, pp.477–492, 2017.
[38] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all You need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Long Beach, CA, USA, pp.6000–6010, 2017.
[39] Y. Z. Zhang, Y. Cao, Y H. Wen, *et al.*, "Optimization of information interaction protocols in cooperative vehicle-infrastructure systems," *Chinese Journal of Electronics*, vol.27, no.2, pp.439–444, 2018.
[40] L. H. Pang, J. Zhang, Y. Zhang, *et al.*, "Investigation and comparison of 5G channel models: From QuaDRiGa, NYUSIM, and MG5G perspectives," *Chinese Journal of Electronics*, vol.31, no.1, pp.1–17, 2022.
[41] H. Duan, T. Feng, S. N. Liu, *et al.*, "Tumor classification of gene expression data by fuzzy hybrid twin SVM," *Chinese Journal of Electronics*, vol.31, no.1, pp.99–106, 2022.



CAO Rui received the B.E. degree in College of Mathematics and Big Data, Anhui University of Science and Technology. She is a Ph.D. candidate of the College of Computer Science and Engineering, Shandong University of Science and Technology. Her research interests include Petri nets, process mining, and deep learning.
(Email: ruicaoqing@163.com)

ZENG Qingtian (corresponding author) received the Ph.D. degree in computer software and theory from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2005. He was a Visiting Professor at the City University of Hong Kong, Kowloon, Hong Kong, in 2008. He is currently a Professor with Shandong University of Science and Technology, Qingdao, China. His current research interests include Petri nets, process mining, and knowledge management.
(Email: qtzeng@sdust.edu.cn)



NI Weijian received the Ph.D. degree in computer science and technology from Nankai University, Tianjin, China, in 2008. He was a Visiting Scholar at the State University of New York, New York, USA, in 2015. He is currently a Professor with Shandong University of Science and Technology, Qingdao, China. His current research interests include process mining, deep learning, and text mining.
(Email: niwj@foxmail.com)

LU Faming received the Ph.D. degree in computer software and theory from Shandong University of Science and Technology, Qingdao, China, in 2013. He is currently a Associate Professor with Shandong University of Science and Technology, Qingdao, China. His current research interests include Petri nets, process mining, and machine learning.
(Email: fm_lu@163.com)

LIU Cong received the Ph.D. degree in the Department of Mathematics and Computer Science, Section of Information Systems (IS), Eindhoven University of Technology, Eindhoven, The Netherlands, in 2019. He is currently a Professor with the Shandong University of Technology, Zibo, China. His current research interests include Petri nets, process mining, and software engineering.
(Email: liucongchina@sdust.edu.cn)

DUAN Hua received the Ph.D. degree in applied mathematics from Shanghai Jiaotong University, Shanghai, China, in 2008. She is currently a Professor with Shandong University of Science and Technology, Qingdao, China. Her current research interests include Petri nets, process mining, and machine learning.
(Email: huaduan59@163.com)