Computing and Informatics, Vol. 44, 2025, 1393–1422, doi: 10.31577/cai_2025_6_1393

CONSTRUCTING AND REPAIRING PROCESS MODELS CONTAINING LOOP RETURN STRUCTURES VIA LOGIC PETRI NETS

Yuyue Du

College of Computer, Shandong Xiehe University
6277 Jiqing Road, Licheng District, 250109 Jinan City, China
e-mail: yydu001@163.com

Wenjing Luan*, Xize Zhang

College of Computer Science and Engineering
Shandong University of Science and Technology
579 Qianwangang Road, Huangdao District, 266590 Qingdao City, China
e-mail: {wenjingmengjing, 15764226785}@163.com

Mei Chen

College of Computer, Shandong Xiehe University
6277 Jiqing Road, Licheng District, 250109 Jinan City, China
e-mail: shuiyuerui@163.com

Abstract. Business processes realized by enterprise information systems are usually designed and verified in advance by process models. Event logs generated from such systems can be used to ensure the correctness of the business process. With the upgrade of systems or changing of customers' requests, it often appears that the actual behaviors observed in event logs are not consistent with those in process models. Process mining techniques are thus used to construct and repair process models by mining event logs. In this work, we propose a new model repair approach based on logic Petri nets (LPNs). It can repair a Petri-net-based process

* Corresponding author

model containing a loop return structure. According to alignments, relations between the activities in event logs and the process models are analyzed such that the deviations can be found. Then, logic expressions of logic transitions are constructed and an LPN-based process model is thus constructed. The repaired model can accurately describe the same processes as the event logs. Finally, some cases related to a thoracic surgery process in a hospital are given. The correctness and effectiveness of the proposed approach are illustrated by experiments.

Keywords: Model repair, process mining, process model containing loop return structures, logic Petri net

Mathematics Subject Classification 2010: 68M20

1 INTRODUCTION

In information systems, business processes are usually designed and verified in advance based on reliable process models. However, the phenomenon that real business processes deviate from the models often appears, which has been confirmed in many studies [1, 2, 3]. Therefore, in order to keep the models consistent with the real business processes, many process mining techniques have been proposed. There are mainly three types of process mining techniques: process discovery, conformance checking, and enhancement.

Process discovery can construct process models (e.g., Petri nets or Business Process Modelling Notation (BPMN) models) from event logs [4, 5, 6, 7, 8, 9, 10]. The idea is simple: suitable process models that describe the behaviors well are constructed by mining event logs. However, choosing suitable process models from such logs is notoriously difficult according to the characteristics of real-life event logs.

Conformance checking is another type of process mining technique [4]. It compares event logs of the same process with an existing process model. It can verify if the model conforms to the real business process and vice versa. Three cases may occur in the process of conformance checking:

1. the process cannot be described by the model where some events in the model are missed;
2. some events that are not described in the models exist in the logs; and
3. the occurrence order of events in the model is different from those in the model.

Therefore, we need to compare a known process model with event logs generated by information systems based on the conformance checking. We can measure the quality of process models based on several metrics, i.e., fitness, simplicity, precision, and generalization [4]. Fitness refers to whether the models can replay a series

of events in the logs, which is one of the most critical quality metrics. If such traces can be replayed by a process model completely, we call that the fitness of the model is ideal. Simplicity requires process models as simple as possible. Precision does not allow activities that are not observed in the logs occur in process models. Generalization refers to that such activities can be reproduced by a model, which may happen in the future.

As a classical process mining algorithm, the α algorithm [11] is proposed based on workflow nets [12, 13]. The differences between observed and modelled events of a model are detected based on conformance checking techniques [4, 14, 15, 16]. There are many extensions of α algorithms. For example, the $\alpha\#$ algorithm [17] is proposed to delete the invisible transitions effectively. However, some process models cannot be effectively mined by the existing methods. Model repair is a new type of process mining algorithm. The original models can be repaired based on model repair techniques, such that they can replay the event logs and hold most characteristics of the reference models. There are many model repair approaches, such as Knapsack's method [18] and Fahland's method [19], which can be used to describe the real processes. To ensure that the process model can conform to the event log, both methods need to add self-loops in the process model. However, lots of unnecessary traces will be generated, which will significantly reduce the accuracy of the repaired model. Besides, it will add the mined sub-processes to one place as loops by collecting some sub-logs, which results in a low precision in the repaired model if such deviations occur in the alignments between the event logs and the process models. The repaired models allow many traces that cannot be observed in the event log. Thus, this study proposes a new model repair approach via logic Petri nets in order to accurately describe the behaviors in event logs. A loop return structure needs to be identified and added. It contains a new activity and a loop body, while the existing approaches cannot identify such structures. The repaired models that are as similar as the original models can replay the actual processes well. Our approach can effectively improve the precision and fitness of the repaired model. This paper has the following contributions:

1. It proposes a method to identify the positions where deviations occur in the model.
2. It proposes a new model repair approach based on a logic Petri net (LPN) by identifying the deviations between the models and the logs. Then, a method is designed to construct the process models containing a loop return structure.
3. It verifies the correctness and effectiveness of our approach by experiments. The results outperform the baseline methods.

The rest of the paper is organized as follows. Section 2 reviews some important definitions. Section 3 introduces our approach for constructing and repairing process models containing loop return structures. Section 4 gives some experimental results. Finally, Section 5 concludes this work.

2 PRELIMINARIES

In this section, the concepts of Petri nets [20, 21], workflow nets [12, 13, 22, 23], alignment [24, 25, 26], and logic Petri nets and some other related concepts are reviewed first.

Definition 1. Let S be a set. A multiset D over S is a function $D : S\mathbb{N}_+$, where $\mathbb{N}_+$ denotes a set of positive integers. $B(S)$ denotes the set of all multisets over S.

For example, $S = \{r, o, h\}$ is a set; $n_1 = \emptyset$ and $n_2 = [o, h, o, h, r]$ are two multisets over S, where n_1 is an empty multiset and for simplicity, n_2 is written as $[r, o_2, h_2]$.

Definition 2. Let $\mathcal{U}c$ be the set of all activities. Given $C \subseteq \mathcal{U}c$, $\sigma \in C^*$ is called a trace; $L \in B(C^*)$ is an event log which contains a set of traces, and $\Phi(\sigma)$ is the set of all activities in trace σ.

Given an event log $L = \{\langle h, g, f, o, p2\rangle^6, \langle f, g, h, o\rangle^4, \langle e, h, d, g, f, p2, o\rangle^3\}$, $\sigma = \langle h, g, f, o, p2\rangle$ is a trace with $\Phi(\sigma) = \{h, g, f, o, p2\}$. 13 cases are included in L, i.e., 6 cases follow the trace $\langle h, g, f, o, p2\rangle$, 4 cases follow the trace $\langle f, g, h, o\rangle$, and 3 cases follow the trace $\langle e, h, d, g, f, p2, o\rangle$. There are $6 \times 5 + 4 \times 4 + 3 \times 7 = 67$ events in total.

Definition 3. $N = (P, T; F)$ is called a net, where P represents a finite set of places, T represents a finite set of transitions, and F represents a set of directed arcs from places to transitions or from transitions to places. For $x \in P \cup T$,

$$^\bullet x = \{y | y \in P \cup T \wedge (y, x) \in F\}, \tag{1}$$

$$x^\bullet = \{y | y \in P \cup T \wedge (x, y) \in F\}, \tag{2}$$

where $^\bullet x$ represents a pre-set of x, $x^\bullet$ represents a post-set of x, and $^\bullet x \cup x^\bullet$ represents the extension of x.

Definition 4. A four-tuple $PN = (P, T; F, M)$ is called a Petri net, where

1. $N = (P, T; F)$ is a net;
2. $M : P \to N_+$ is called a marking function; and
3. Transition firing rules:
   (a) For $\forall t \in T$, if $\forall p \in {}^\bullet t : M(p) \geq 1$ can be enabled denoted by $M[t >$; and
   (b) If $M[t >$, t will be fired and a new marking M' is generated, denoted by $M[t > M'$, where

$$M'(p) = \begin{cases} M(p) - 1, & p \in^\bullet t - t^\bullet, \\ M(p) + 1, & p \in t^\bullet -^\bullet t, \\ M(p), & \text{else.} \end{cases} \tag{3}$$

$R(M)$ represents the set of all reachable markings from M, and $M \in R(M)$.

There are four structures in Petri nets, i.e., parallel, loop, sequence and choice structures. The initial and final transitions are the same in each branch in a parallel structure of a PN. For convenience, we use t_{ci} and t_{cf} to represent an initial transition and a final transition of a parallel structure, respectively.

Definition 5. Let $PN = (P, T; F, M)$ be a Petri net and for i, $o \in P$, we have ${}^\bullet i = \varnothing$ and $o^\bullet = \varnothing$. $WFN = (PN, i, o)$ represents a workflow net where i represents an initial place and o represents a final place. For $\forall x \in P \cup T$, there always exists a path from i to o.

Definition 6. Let $\sigma \in C^*$ be a log and $PN = (P, T; F, M)$ be a Petri net. A move is a 2-tuple $(c, t) \in \{C^{>>} \times T^{>>}\} \setminus \{(>>, >>)\}$, where $>>$ represents no move, $C^{>>}$ represents all the activities and $>>$ in the log, and $T^{>>}$ represents all the transitions and $>>$ in the model, where

1. (c, t) is a log move if $c \in C$ and $t = >>$;
2. (c, t) is a model move if $t \in T$ and $c = >>$;
3. (c, t) is a synchronous move if $t \in T$ and $c \in C$; and
4. Otherwise, (c, t) is call an illegal move.

Definition 7. Let $PN = (P, T; F, M)$ be a process model. m' is a reachable marking from m if there exists a sequence $\sigma \in T^*$ such that $m[\sigma > m'$. A sequence $\sigma \in T^*$ is called a complete firing sequence if $m_i[\sigma > m_f$, where $m_i = \{i\}$ is an initial marking and $m_f = \{o\}$ is a final marking.

Definition 8. Given $\sigma \in C^*$ and $PN = (P, T; F, M)$, a likelihood cost function for different move is denoted by $lc : (C^{>>} \times T^{>>} \to IR)$, IR represents a set of all non-negative real values. A move sequence $\gamma \in (C^{>>} \times T^{>>})^*$ is called an alignment between σ and PN, where symbol $\pi_1(\gamma)_{\downarrow C}$ denotes the move sequence of γ in the trace and $\pi_2(\gamma)_{\downarrow T}$ denotes the move sequence of γ in the model. $\pi_1(\gamma)_{\downarrow C} = \sigma$, i.e., its move sequence in the trace (ignoring $>>$) can generate the trace sequence; and $\pi_2(\gamma)_{\downarrow T}$ represents that its move sequence in the model can generate a completing firing sequence of PN. The alignment $\gamma \in (C^{>>} \times T^{>>})^*$ is an optimal alignment if $\forall \gamma' \in \Phi_{\sigma,PN} \colon \Sigma_{(c,t)\in\gamma} lc((c, t)) \leq \Sigma_{(c',t')\in\gamma'} lc((c', t'))$, where $\Phi_{\sigma,PN}$ represents the set of all alignments between σ and PN.

Next, we illustrate the above definitions with an example.

Example 1. A simple Petri net N_1 is shown in Figure 1. Let $\sigma_1 = \langle a, f, c, d, g\rangle$ be a trace. The optimal alignment between σ_1 and N_1 is shown in Figure 2, i.e., $\gamma_1 = \langle (a, a), (f, f), (>>, b), (c, c), (d, >>), (g, g)\rangle$, where (g, g) is synchronous moves, $(d, >>)$ is a log move and $(>>, b)$ is a model move.

Definition 9. Let $\mathcal{U}c$ be the set of all activities, $C \subseteq \mathcal{U}c$, and $\sigma \in C^*$. Let $\oplus = \{\otimes, \rightarrow, \circlearrowleft, \wedge\}$ be a set of operators. A process tree denoted by PT is constructed as follows:

1. $c \in C \cup \{\tau\}$ represents a process tree; and
2. If there are two process trees d and e, then $\oplus(d, e)$ represents a process tree, where
   (a) $\wedge(d, e)$ represents the parallel structure of d and e, i.e., the two process trees d and e need to be fired synchronously;
   (b) $\circlearrowleft(d, e)$ represents the loop structure of d and e, i.e., the two process trees d and e can be fired infinitely and in no order;
   (c) $\rightarrow(d, e)$ represents the sequential structure of d and e; and
   (d) $\otimes(d, e)$ represents the choice structure between d and e.

Definition 10. $LPN = (P, T; F, I, O, M)$ is an logic Petri net (LPN) where P is a finite set of all places; $T = T_I \cup T_O$ is a finite set of all transitions with $T \cap P = \emptyset$, $T \cup P \neq \emptyset$, $t \in T$ and ${}^\bullet t \cap t^\bullet = \emptyset$, where T_I is a set of logic input transitions, and for $\forall t \in T_I$, a logic input function $f_I(t)$ restricts the set of input places ${}^\bullet t$, and T_O is a set of logic output transitions, and for $\forall t \in T_O$, a logic output function $f_O(t)$ restricts the output places $t^\bullet$; $F \subseteq (T \times P) \cup (P \times T)$ is a set of arcs; I is a logic input mapping: for $\forall t \in T_I$, $I(t) = f_I(t)$; O is a logic output mapping: for $\forall t \in T_O$, $O(t) = f_O(t)$; $M{:}P \rightarrow \{0, 1\}$ represents a marking function, and the number of tokens in p are denoted by $M(p)$; and transition firing rules are given as follows:

1. If $f_I(t)|_M = .T.$, $\forall t \in T_I$ can fire where for $\forall p \notin t^\bullet \cup^\bullet t$, $M'(p) = M(p)$; for $\forall p \in t^\bullet$, $M'(p) = 1$; for $\forall p \in {}^\bullet t$, $M'(p) = 0$; and
2. If $\forall p \in^\bullet t$, $M(p) = 1$, $\forall t \in T_O$ can fire where $\forall p \in t^\bullet$ must satisfy $f_O(t)|_{M'} = .T.$ and $\forall p \notin t^\bullet \cup^\bullet t$, $M'(p) = M(p)$, $\forall p \in^\bullet t$, $M'(p) = 0$.

We present a simple logic Petri net LN_1 in Figure 3, where b is the logic output transition, $\mathrm{O}(b) = (p_3 \wedge p_4) \otimes p_5$, and $I(f) = (p_6 \wedge p_7) \otimes p_8$. When b fires, it results in two cases:

1. both p_3 and p_4 get a token; and
2. p_5 contains a token.

When f is enabled, two conditions are satisfied:

1. both p_6 and p_7 contain a token; and
2. p_8 contains a token.

3 CONSTRUCTING AND REPAIRING PROCESS MODELS CONTAINING LOOP RETURN STRUCTURES

We propose an approach for constructing and repairing the models with loop return structures based on LPNs in this section. Some relevant definitions will be pre-

sented first. We then describe our proposed approach, as well as two model repair approaches.

Definition 11. Given a log $L \in B(C^*)$, if $a \in L$, $t \in T$, then $\eta(a,t)$ refers to the activity a in the logs corresponds to the transition t in the process models.

When the traces in the event logs contain activities that do not exist in the model, we need to find out the new activities by projecting the activities in event logs to the transitions in process models.

A Petri net model N_2 with a parallel structure is shown in Figure 4 in Example 1, and $L_1 = \{\langle c,d,e,f,g,h,a,b\rangle, \langle d,c,b,a,h,e\rangle, \langle d,c,b,a,e,f,g,h\rangle\}$.

A Petri net model N_2 is shown in Figure 4. Figure 5 shows the process tree of N_2 denoted by PT_1. According to Definition 9, we can get the formal description of the process tree PT_1, i.e., $PT_1 = \rightarrow (a, b, \wedge(c,d,e), f, g)$.

Definition 12. Let $PN = (P, T; F, M)$ and $C \in \mathcal{U}c$ is the process tree of PN and n is a node of PT. If the sub-node set of n is N_S and $n_s \in N_S$, a tuple (t_{ci}, t_{cf}) is called a concurrent initial-final pair, where $n{=}\wedge$ and $\exists n \in PT$, $t_{ci} = {}^\bullet({}^\bullet(n_s))$, and $t_{cf} = ((n_s)^\bullet)^\bullet$.

S_{CIFP} denotes all concurrent initial-final pairs in a Petri net, i.e., $S_{CIFP} = \{(t_{ci}, t_{cf}) \mid t_{ci} = {}^\bullet({}^\bullet(N_S)), t_{cf} = ((N_S)^\bullet)^\bullet, \forall n \in PT\}$ and $n = \wedge$.

For a process tree, we can give a formal translation from each of the operators to the Petri net. The process tree provides information about transitions and structures of the process model. We traverse the process tree to find out the nodes of the parallel structure. According to the information of all the nodes, we obtain all the transitions of the parallel structures. Then, we get the initial and final transitions of all the parallel structures. If there is a node with "$\wedge$" in a process model, then there is a parallel structure in the corresponding process model. We get all the sub-nodes, e.g., c, d and e are the sub-nodes of the node "$\wedge$" in Figure 5. A concurrent initial-final pair is obtained according to the above definition. Then the set of concurrent initial-final pairs are constructed by collecting all concurrent initial-final pairs in the Petri net. We will give the formal definition as follows.

Definition 13. Given an event log $L{\in}B(C^*)$, and $\sigma \in L : o$, $p{\in}\Phi(\sigma)$. We can obtain the ordering relations between o and p as follows:

1. Direct follow relation $>$: $o > p$ if $\sigma = \langle a_1, a_2, a_3, \ldots, a_n\rangle$, $i{\in}\{1, \ldots, n{-}1\} : \sigma \in L$, $o = a_i$ and $p = a_{i+1}$;
2. Indirect follow relation $>>_r$: $a_1 >>_r a_n$ if $\exists a_1, a_2, a_3, \ldots, a_n \in \Phi(\sigma) : a_1 > a_2 > \cdots > a_n$;
3. Choice relation $\# : o\#p$ if $\sigma{\in}L{:}o{\in}\Phi(\sigma)$ and $p \notin \Phi(\sigma)$ or $p \in \Phi(\sigma)$ and $o \notin \Phi(\sigma)$;
4. Direct casual relation $\rightarrow_D$: $o \rightarrow_D p$ if $\sigma \in L$, $o, p \in \Phi(\sigma) : o > p$ and no $p > o$;
5. Indirect casual relation $\Rightarrow$: $a_1 \Rightarrow a_n$ if $\exists a_1, a_2, a_3, \ldots, a_n \in \Phi(\sigma) : a_1 \rightarrow_D a_2 \rightarrow_D \cdots \rightarrow_D a_n$;

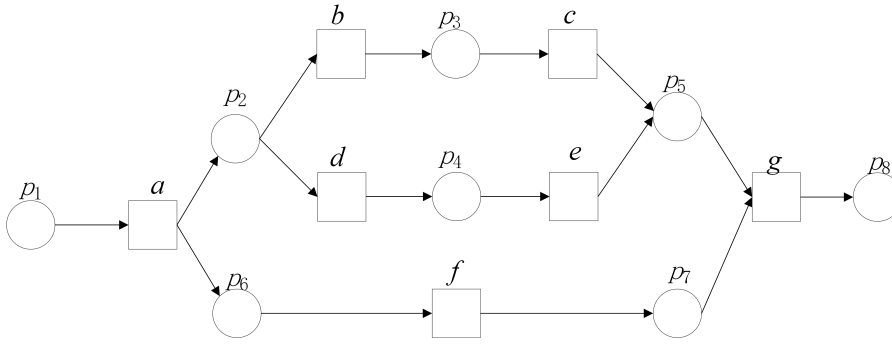


Figure 1. A simple Petri net N_1

$$\gamma_1 = \begin{array}{|c|c|c|c|c|c|} \hline a & f & >> & c & d & g \\ \hline a & f & b & c & >> & g \\ \hline \end{array}$$

Figure 2. The optimal alignment γ_1 between σ_1 and N_1

6. Concurrent relation $||$: $o||p$ if σ, $\sigma' \in L$ for $o, p \in \Phi(\sigma)$: $o > p$ and for $o, p \in \Phi(\sigma')$: $p>o$.

Definition 14. There are two transitions m and n, and two traces $\sigma, \sigma' \in L$. m and n are concurrent transitions if $n, m \in \Phi(\sigma) : n > m$ and $n, m \in \Phi(\sigma') : m > n$.

We identify the concurrent structures based on the above definitions. Each branch of the concurrent structures can be treated as a whole while each branch may contain other structures. Each branch represents a path from the start place to the end place of the concurrent structures. By using Algorithm 1 to identify the concurrent structures, we can get the set of concurrent transitions and the set of concurrent initial-final pairs.

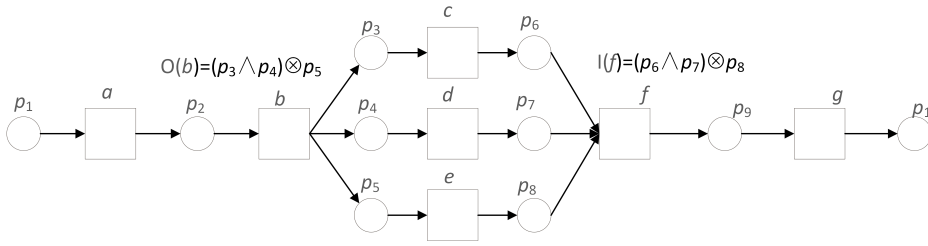


Figure 3. A simple logic Petri net model LN_1

Algorithm 1: Computation of the set of concurrent transitions S_C and the set of concurrent initial-final pairs S_{CIFP}

Input: $PN = (P, T; F, M)$, and a non-leaf node of PT represented by N_{nl}
Output: The set of concurrent transitions denoted by SC and the set of concurrent initial-final pairs denoted by S_{CIFP}

$S_{CIFP} \leftarrow \emptyset$, $S_C \leftarrow \emptyset$;
If ($N_{nl} \in \oplus$ and $N_{nl}! = \emptyset$ and $N_{nl} =$ "$\wedge$" and $N_S \in N_{nl}$) Then
$S_{CIFP} \leftarrow S_{CIFP} \cup \{{}^{\bullet}({}^{\bullet}(N_S)), ((N_S)^{\bullet})^{\bullet}\}$;
Else
For all the sub-nodes $n_s \in N_s$ Do
S_{CIFP} (node, PN);
$S_C \leftarrow S_C \cup \{n_s\}$;
Return S_{CIFP} and S_C.

Next, we give the definition of the projection as follows.

Definition 15. Let S_T be a set of transitions, and $D \subseteq S_T$. For $\sigma_i \in D^*$, the projection of σ_i on D is denoted by $\sigma_i|_D = \{t | t \in D \wedge t \in \sigma_i\}$.

For example, $bbaca|_{\{a,b\}} = bbaa$. From Definition 1, we have $[e^4, f, g^3]|_{\{e,g\}} = [e^4, g^3]$.

Example 2. A Petri net N_2 with a parallel structure is shown in Figure 4, and an event log is $L_3 = \{\langle a, b, c, f, g, h, d, e, f, g\rangle, \langle a, m, b, d, f, g, h, d, e, f, g\rangle\}$. Figures 6 and 7 show the two optimal alignments between N_2 and the two traces of L_3, which are denoted by γ_2 and γ_3, respectively.

There are multiple log moves and model moves in Figures 6 and 7. We see that two new activities and multiple repeated activities have been added to the event log L_3. There are many deviations in the two traces of the log L_3, and it can be found that the branches of the concurrent structure are repeatedly fired after a new activity h is fired, i.e., both traces have loop structures. However, Fahland's method [19] cannot find this structure and handle the repeated activities correctly.

Next, the alignment of the logic Petri net will be introduced for model evaluation and analysis. Compared with the traditional alignment, the alignment of LPN is similar. The difference is the change of logic transition firing rules in the model. We now give the formal definition below.

Definition 16. Given $\sigma \in C^*$ and $LPN = (P, T; F, I, O, M)$. A move is denoted by the two tuple $(c, t) \in C^{>>} \times T^{>>} \backslash \{(>>, >>)\}$, where $>>$ represents no move in either the model or the log. A logic alignment $\gamma \in (C^{>>} \times T^{>>})^*$ represents a move sequence between the model LPN and a trace, where $\pi_1(\gamma)_{\downarrow C} = \sigma$ represents that the move sequence in the trace (ignoring $>>$) can generate the trace sequence based on the logic alignments; and $\pi_2(\gamma)_{\downarrow T}$ represents its move sequence in the model that can generate a completing firing sequence of LPN based on the logic transition firing rules.

Definition 17. Let $\sigma \in C^*$ and $LPN = (P, T; F, I, O, M)$, and a likelihood cost function for different move is denoted by $lc : (C^{>>} \times T^{>>}) \to IR$. A logic alignment $\gamma \in (C^{>>} \times T^{>>})^*$ representing a move sequence between LPN and σ is an logic optimal alignment if $\forall \gamma' \in \Phi_{\sigma,LPN} : \Sigma_{(c,t)\in\gamma} lc((c,t)) \leq \Sigma_{(c',t')\in\gamma'} lc((c',t'))$, where $\Phi_{\sigma,LPN}$ represents the set of all logic alignments between σ and LPN.

Definition 18. Cost function δ: $C \to N$ assigns costs to model moves and log moves, which N represents the set of positive integers. $\delta(\gamma) = \Sigma_{(c,t)\in\gamma}\delta((c,t))$ can be used to denote the cost of an alignment $\gamma \in (C^{>>} \times T^{>>})^*$.

There may also be multiple logic optimal alignments in the set of all the logic alignments. Likewise, the likelihood cost functions for log moves and model moves are $lc(a, >>) = lc(>>, t) = 1$, and for synchronous moves, $lc(a, t) = 0$.

Example 3. An LPN model N_{L1} is shown in Figure 8, and $L_4 = \{\langle a, b, c, d, e\rangle, \langle a, d, e, e\rangle\}$. Figures 9 and 10 show the two logic optimal alignments denoted by γ_{L1} and γ_{L2}.

As shown in Figure 9, the first trace σ_1 in L_4 can replay the logic model N_{L1} perfectly, i.e., there is no log move or model move in the alignment. There are multiple logic optimal alignments based on the second trace σ_2 of L_4 and the model N_{L1}, and only one is presented here.

As can be seen from Example 3, the replay of the traces in the event logs needs to be controlled according to the logic functions of the logic transitions based on the logic Petri net. In Figure 10, the likelihood cost functions satisfy that $lc(\gamma_{L2}) = lc(a, >>) + lc(>>, t) = 2$, and the logic alignment is the lowest cost alignment, which is a logic optimal alignment. The formal definitions of the loop sub-trace and the loop sub-log is introduced next.

Definition 19. Let $C \subseteq \mathcal{U}c$ be a set of activities, $\sigma \in C^*$ be a trace on C, and $PN = (P, T; F, M)$ be a Petri net. γ represents an optimal alignment between the trace σ and the model PN. The loop sub-trace ST is the sequence of many continuous log moves $(a_i, >>), \ldots, (a_j, >>)$ in the optimal alignment. $L_{ST} = (a_i, \ldots, a_j)$, and a_i does not exist in the original model. but other activities have appeared in the original model. A multiset $L_{SL} \in B(C^*)$ can be called a loop sub-log, which represents a set of the loop sub-traces. The loop sub-log contains the start and end activities of the loop body.

In the case that there is a loop in the updated log, in order to find the loop structure more accurately, we first need to collect the loop sub-log. Then, we need to locate the deviations according to different log moves, and finally repair the model. In order to collect the loop sub-log, log-based ordering relations need to be obtained among the activities in the log.

We present a log-based ordering relation determination algorithm, as shown in Algorithm 2, aimed at obtaining log-based ordering relations and collecting the loop sub-log, where RL represents the set of all the log-based ordering relationships.

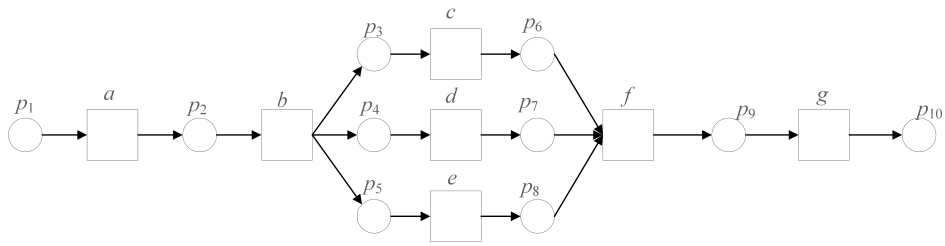


Figure 4. A model N_2 containing a parallel structure

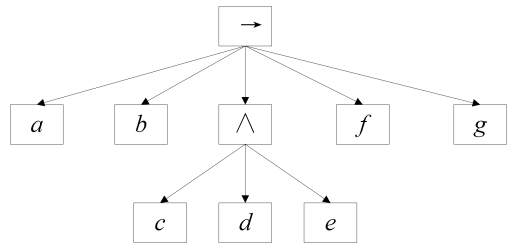


Figure 5. The process tree PT_1 of N_2

$\gamma_2 =$

a	b	c	f	g	h	d	e	f	g
a	b	c	f	g	>>	>>	>>	>>	>>

Figure 6. The optimal alignment of σ_1 and N_2

$\gamma_3 =$

a	m	b	d	f	g	h	d	e	f	g
a	>>	b	d	f	g	>>	>>	>>	>>	>>

Figure 7. The optimal alignment of σ_2 and N_2

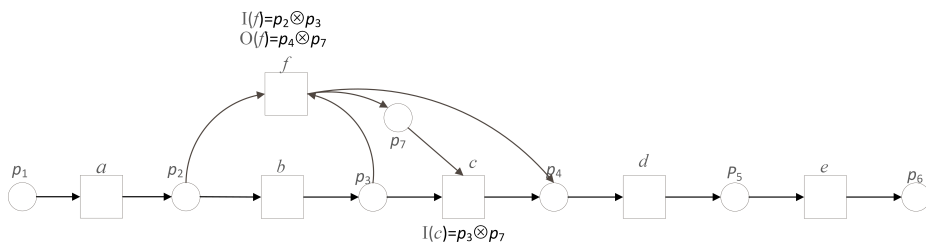


Figure 8. A logic Petri net model N_{L1}

$\gamma_{L1} =$

a	b	c	d	e
a	b	c	d	e

Figure 9. The logic optimal alignment of σ_1 and N_{L1}

$\gamma_{L2} =$

a	$>>$	d	e	e
a	f	d	e	$>>$

Figure 10. The logic optimal alignment of σ_2 and N_{L1}

In Algorithm 2, steps 8, 10 and 12 are executed in parallel to save time. The relations are obtained by using Algorithm 2. In Algorithm 3, *count* represents the number of traces in the log, σ_{e1} represents the $e1$ trace in the log, σ_e represents the e trace in the log, and σ_f represents the f trace in the log.

For the event log L_3 in Example 2, $\{(h, >>), (d, >>), (e, >>), (f, >>), (g, >>)\}$ is a sequence of log moves. Since f and g have been replayed before, we have $c||d$ and $c||e$. h is then added to the loop as a new activity that does not exist in the original model. We construct the loop sub-log as $\{\langle h, d, e, f, g\rangle\}$.

After collecting the loop sub-log, it is necessary to locate the deviation positions in order to repair the model. First, we introduce the following definition.

Definition 20. Let $C \subseteq \mathcal{U}c$, $\sigma \in C^*$ and $PN = (P, T; F, M)$. γ represents an optimal alignment between σ and PN. L_{ST} is a loop sub-trace, which is a loop sub-log. For $p \in P$, the set of its preorder places is denoted by $P_{bef|p} = \{p | p \in {}^\bullet t_i \wedge t_i \in L_{SL}\}$, and $|P_{bef|p}|$ denotes the number of the elements.

Definition 21. Let $C \subseteq \mathcal{U}c$, $\sigma \in C^*$ and $PN = (P, T; F, M)$. γ represents an optimal alignment between σ and PN. If there is any model movement or log movement, it indicates that there is a deviation, and $D_L(p_i, p_o)$ represents the deviation positions, where p_i and p_o are the initial and end places of the deviation positions, respectively.

Definition 22. Let $L \in B(C^*)$, $PN = (P, T; F, M)$, γ be the optimal alignment between L and PN, and $P_{bef|p}$ be the set of preorder places of $p \in P$. $\exists a_i, \ldots, a_j \in L_{SL}$, and $P_L = \{p | p \in p_{bef|p}, t \in {}^\bullet p \wedge t \notin L_{SL}\}$, if $|P_L| > 1$, P_L is called the set of preorder places of the concurrent structure.

The set of preorder places mainly aims at finding the post-set places of the loop transitions for the loop structures with concurrent relations. $P_{bef|p}$ includes all the pre-set places in the loop sub-log. Next, we give an algorithm to determine the deviations, as shown in Algorithm 4.

Algorithm 2: Generation of log-based ordering relations

Input: A complete event log $L \in B(C^*)$
Output: The set R_L

$R \leftarrow \varnothing$, $R_L \leftarrow \varnothing$;
For $(\sigma_m \in L : m_i \in \sigma_m, i=1;;i++)$ Do
$R \leftarrow R \cup \{m_i > m_{i+1}, m_i >>_r m_{i+j}\}$, $j = 2, \ldots, |\sigma| - i$;
If ($v \in \sigma_m$ and $w \notin \sigma_m$ OR $w \in \sigma_m$ and $v \notin \sigma_m$) Then
$R_L \leftarrow R_L \cup \{v \# w\}$;
If ($w \not> v$ and $v > w$) Then
$R_L \leftarrow R_L \cup \{v \rightarrow w\}$;
If ($w \rightarrow q$ and $v \rightarrow w$) Then
$R_L \leftarrow R_L \cup \{v \Rightarrow q\}$;
If ($v \rightarrow w$ and $w \Rightarrow d$) Then
$R_L \leftarrow R_L \cup \{v \Rightarrow d\}$;
If ($v \Rightarrow q$ and $q \Rightarrow e$) Then
$R_L \leftarrow R_L \cup \{v \Rightarrow e\}$;
If ($v > w$ and $w > v$) Then
$R_L \leftarrow R_L \cup \{v || w\}$;
Return R_L.

Algorithm 4 is to determine the deviation positions. By Algorithm 4, we get all the post-set places of the loop transition a_i. If there is a concurrent structure in the loop structure, the set of preorder places of the activities in the loop sub-log is first collected, and then the pre-set transitions of all the preorder places are found. If the transition is not in the loop sub-log, we need to know the number of the post-set places of the transition. If it is greater than 1, we know that the post-set places

Algorithm 3: Compute the loop sub-log L_{SL}

Input: The optimal alignment γ between the log $L \in B(C^*)$ and the process model PN, and $PN = (P, T; F, M)$
Output: The loop sub-log L_{SL}

$L_{SL} \leftarrow \emptyset$;
Traversing the optimal alignment γ, if $a_i \in A, t_i =>>$, and there are many continuous log moves $\{(a_i, >>), \ldots, (a_j, >>)\}$;
For $(\sigma_m \in L: m_i \in \sigma_m, i=0;;++i)$ Do
If ($a_i \notin T, a_{i+1}, \ldots, a_j \in \sigma_{e1}, 1 <= e1 <= count$) Then
$L_{SL} \leftarrow L_{SL} \cup \{a_i, \ldots, a_j\}$;
Else If ($a_{i+k} \notin \sigma_e, a_{i+k} \in \sigma_f$ and $b_{i+k} \in \sigma_e, b_{i+k} \notin \sigma_f, 1 <= k <= j - i$) Then
If ($b_{i+k} \# a_{i+k}$ or $b_{i+k} \| a_{i+k}$) Then
$L_{SL} \leftarrow L_{SL} \cup \{a_{i+k}\}$;
Return L_{SL}.

Algorithm 4: Locating the deviation positions $D_L(p_i, p_o)$

Input: The optimal alignment γ between $L \in B(C^*)$ and $PN = (P, T; F, M)$, and the set of preorder places of $p \in P$ denoted by $P_{bef|p}$

Output: the deviation positions $D_L(p_i, p_o)$

$D_L(p_i, p_o) \leftarrow \emptyset$;
Using Algorithm 3 to collect the loop sub-log L_{SL}, and traverse the optimal alignment γ, and if $a_i \in A, t_i =>>$;
If $(\exists\{a_i, \ldots, a_j\} \in L_{SL}$, and $P_L = \{p \mid p \in P_{bef|p}, t \in^\bullet p \wedge t \notin L_{SL}\})$ Then
If $(|P_L| > 1)$ Then
$p_i \leftarrow a^\bullet_{i-1}$, $p_o \leftarrow P_L$;
If ($a_i \notin T$ and $a^\bullet_{i-1} =^\bullet a_{i+1}$) Then
$p_i \leftarrow p$, $p_o \leftarrow^\bullet a_{i-1}$;
Else
$p_i \leftarrow a^\bullet_{i-1}$, $p_o \leftarrow^\bullet a_{i+1}$;
If ($a_i \in T$) Then
$p_i \leftarrow a^\bullet_i$, $p_o \leftarrow^\bullet a_i$;
For the model move $a_i =>>, t_i \in T$ Do
$p_i \leftarrow^\bullet a_i$, $p_o \leftarrow a^\bullet_i$;
Return $D_L(p_i, p_o) \leftarrow (p_i, p_o)$.

corresponding to the transition are the pre-set places of the concurrent structure. All the pre-set places of the parallel structure can be found in this way, and then the deviations are identified.

In Example 2, Figure 6 shows the optimal alignments denoted by γ_2 between the first trace of L_3 with deviations and N_2. We see that the loop sub-log based on the first trace is $\{\langle h, d, e, f, g\rangle\}$. The deviation position is obtained and denoted as $(p_{10}, (p_4, p_5))$.

Definition 23. Given $PN = (P, T; F, M)$, $t_p \in T$ is the preorder transition of t if $p \in^\bullet t$ and $t_p \in^\bullet p$. The set of logic preorder transitions is denoted by $T_{bef|t} = \{t_p | p \in^\bullet t$ and $t_p \in^\bullet p\}$.

Definition 24. Given $PN = (P, T; F, M)$, for $t \in T$, $T_{bef|t}$ denotes the set of logic preorder transitions. $(\boldsymbol{AI}|t)[m_2][m_2](m_2 = |T_{bef|t}|)$ represents the logic preorder matrix. For $0 <= o$, $p <= m_2$, if $o \neq p$, then $(\boldsymbol{AI}|t)[o][p] = lr(o, p)$; otherwise, $(\boldsymbol{AI}|t)[o][p] = \emptyset$.

In [27], the logic relation $lr(r, c)$ is defined as follows based on the above definition:

1. $r \otimes c : T_{bef|t}$ allows r or c to be fired when t is fired;
2. $r||c : T_{bef|t}$ allows both r and c will be fired when t is fired, but they cannot happen at the same time;

3. $r \vee c$. $T_{bef|t}$ allows both r and c can be fired when t is fired; and
4. $r \wedge c$. r and c will be fired simultaneously when t is fired.

$I(t) = pl(\lambda)$ can be constructed based on the logic preorder matrix $(\boldsymbol{AI}|t)[m][m]$ $(m = |T_{bef|t}|)$, and the preorder transition function $\lambda(T', \boldsymbol{AI}|t)$ will be obtained. Next, we introduce the approach in Algorithm 5 for constructing and repairing the models with loop return structures based on LPNs.

Algorithm 5: The approach for constructing and repairing the models with loop return structures based on LPNs

Input: $PN = (P, T; F, M)$ and the optimal alignment γ between the log $L \in B(C^*)$ and PN

Output: The repaired model, i.e., $LPN''' = (P''', T'''; F''', I''', O''', M''')$

$N \longleftarrow LPN'''$, $D_L(p_i, p_o) \leftarrow \emptyset$, $L_{SL} \leftarrow \emptyset$;
Use Algorithm 2 to get R_L, use Algorithm 3 to collect L_{SL} and use Algorithm 4 to get $D_L(p_i, p_o)$;
If $(a_i =>>, t_i \in T$ and $D_L(p_i, p_o) = ({}^\bullet a_i, a_i^\bullet)$ Then
$T''' \leftarrow T''' \cup \tau, F''' \leftarrow F''' \cup \{({}^\bullet a_i \rightarrow \tau) \cup (\tau \rightarrow a_i^\bullet)\}$;
Else if $(a_i \in A, t_i =>>)$ Then
If $(\exists D_L(p_i, p_o) = (p, a_{i-1}^\bullet))$ Then
$T''' \leftarrow T''' \cup a_i, P''' \leftarrow P''' \cup \{p\}, F''' \leftarrow F''' \cup \{(a_{i-1} \rightarrow p) \cup (p \rightarrow a_i)\} \cup (a_i \rightarrow a_{i-1}^\bullet)\}$;
If $(\exists D_L(p_i, p_o) = (a_i^\bullet, {}^\bullet a_i))$ Then
$T''' \leftarrow T''' \cup \tau, F''' \leftarrow F''' \cup \{(a_{i-1}^\bullet \rightarrow \tau) \cup (\tau \rightarrow {}^\bullet a_i)\}$;
If $(\exists D_L(p_i, p_o) = (a_{i-1}^\bullet, {}^\bullet a_{i+1}))$ Then
$T''' \leftarrow T''' \cup \{a_i\}, F''' \leftarrow F''' \cup \{(a_{i-1}^\bullet \rightarrow a_i) \cup (a_i \rightarrow {}^\bullet a_{i+1})\}$;
Use Algorithm 1 to get S_{CIFP} and S_{CT};
If $((t_{cs}, t_{ce}) \subseteq S_{CSEP}, t_j, t_k \in S_{CT}$ and $t_{cs}^\bullet ==^\bullet t_j, t_{cs}^\bullet =^\bullet t_k)$ Then
$I''' \leftarrow I''' \cup \{I(a_i) = \wedge(a_{i-1}^\bullet \otimes a_{i-1}^\bullet)\}$;
$O''' \leftarrow O''' \cup \{O(a_i) =^\bullet t_j \wedge^\bullet t_k, O(t_{ce}) = (t_j^\bullet \wedge t_k^\bullet) \vee (\wedge(P_{bef|}t_{ce}^\bullet))\}$;
Return $LPN''' = (P''', T'''; F''', I''', O''', M''')$.

The repaired models containing loop return structures can be obtained by the aforementioned algorithms. Algorithm 5 consists of some main steps written as Algorithms 1, 2, 3, and 4. First the set of new activities A_{new} are obtained. We then identify the parallel structures, and get the set of concurrent initial-final pairs S_{CIFP} by Algorithm 1. The relations are obtained by Algorithm 2. We collect the sub-log L_{SL} by Algorithm 3. Next, the deviations are located by Algorithm 4. Finally, we construct and repair the models with loop return structures based on LPNs by Algorithm 5.

Example 4. Figure 11 shows a Petri net model N_3, and $L_5 = \{\langle a, b, d, f, l, p1, d, f, l, m, o\rangle, \langle a, b, d, f, l, m, p1, d, f, l, m, o\rangle, \langle a, b, d, f, l, p1, e, g, l, n, o\rangle, \langle a, b, d, f, l, m, p1, e,$

$g, l, m, o\rangle\}$. Figures 12, 13, 14, and 15 show four optimal alignments denoted respectively by $\gamma_4 - \gamma_7$ of the four traces of L_5 with deviations and the model N_3.

We can get the new activity p, the concurrent initial-final pair (a, l), and the loop sub-log $L_{SL1} = \{\langle p1, d, f, l\rangle, \langle p1, d, f, l, m\rangle, \langle p1, e, g, l\rangle, \langle p1, e, g, l, m\rangle\}$. We have $d\#e$, $f\#g$, and $D_L(p_i, p_o) = ((p_8, p_9), (p_3, p_4))$. The model is repaired by using Algorithm 5 as shown in Figure 16. The repaired models, which are based on Knapsack's method [18] and Fahland's method [19], are shown in Figures 17 and 18, respectively.

The idea of Fahland's method is to use an alignment approach for obtaining the deviations. The existing process mining algorithms are used to collect the sub-logs and mine the sub-processes. A self-loop structure is added for repairing the sub-logs. The idea of Knapsack's method [18] is that an alignment approach is used to identify the deviations between model moves and log moves. The sub-processes are mined based on existing methods, and the sub-logs are collected based on these deviations. The original model adds a self-loop structure for repairing the sub-logs. Besides, the cost of alignment needs to be considered in Knapsack's method [18]. Many different repair schemes are compared by setting the cost of log moves and model moves separately. Knapsack's method [18] can obtain the 'best' model repair approach by computing the minimum total cost of log moves and model moves. We know that the repaired model contains no invisible transitions based on Knapsack's method [18]. Besides, the accuracy of the model will be reduced significantly based on both methods. Table 1 presents a comparison of added elements in Example 4 via the three approaches, including the number of added repetitive transitions (ADC)$|T|$, the number of added transitions (ATC)$|T + \tau|$, the number of added flow relations (AFC)$|F|$, and the number of added places (APC)$|P|$, respectively.

The model can be repaired by Fahland's method very poorly, and the method needs to add many repetitive activities, transitions, flow relations and places. The repaired model is very different from the original model. Figure 18 shows that many self-loops are added. Besides, the accuracy of the model will be reduced based on such methods significantly.

The Knapsack's method also needs to add many repetitive activities, transitions, and flow relations in the original model. Figure 17 shows that many self-loops are added in the repaired model. However, such sub-processes may not be allowed to repeat in the real process, because many useless traces will generate, and the accuracy of the model will be reduced significantly.

The total complexity of Algorithms 1, 2, 3, and 4 determine the complexity of Algorithm 5. In Algorithm 5, the relations between the transitions in the choice structure are obtained by using Algorithm 2, and the time complexity is $O(n)$; the loop sub-log L_{SL} is collected by using Algorithm 3, and the time complexity is $O(n)$; the deviations are located by using Algorithm 4, and the time complexity is $O(n)$; Step 12 is to use Algorithm 1 to get the set of concurrent transitions S_C and the set of concurrent initial-final pairs S_{CIFP}, and the time complexity is $O(n)$. Thus, we know that the time complexity of Algorithm 5 is $O(n)$. As a result, it is efficient for

our approach to construct and repair the models with loop return structures based on LPNs.

We project the activities in the event logs onto the transitions in the model, and then the new activities are obtained. We get the log moves and the model moves based on the optimal alignment. Then, the relation between these different activities and the new activities are obtained by using Algorithm 2. We collect the loop sub-log by using Algorithm 3. The deviation positions are located by using Algorithm 4. The new activities are regarded as the loop return transitions. We get the places that need to be controlled. The logic input and output functions are constructed by using Algorithm 5. The actual process can be described by the repaired model, which is obtained by our approach correctly. The repaired models are as similar as possible to the original models. The precision and the fitness of the repaired model are improved by our approach effectively. Therefore, we discuss the weakness of the existing methods for obtaining process models containing loop return structures. For this condition, we find out all the loop return transitions. Then an improved method for repairing such models is proposed.

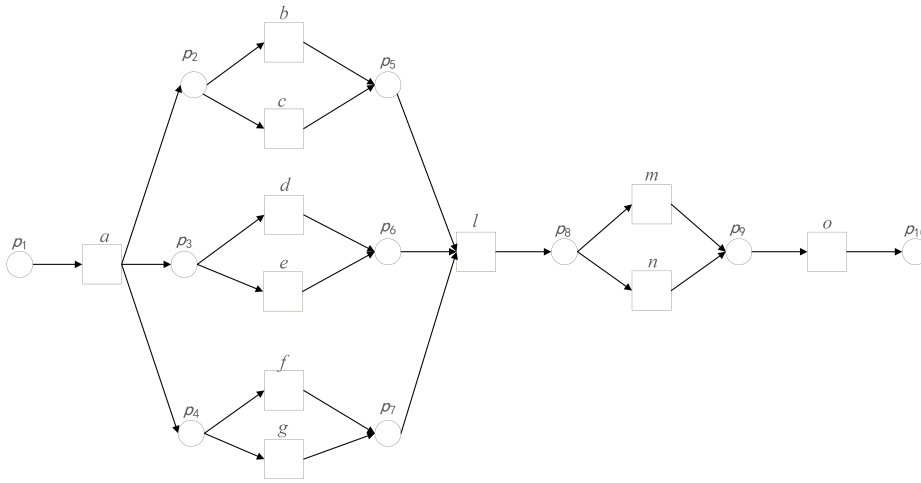


Figure 11. A Petri net model N_3

$\gamma_4 =$

a	b	d	f	l	$p1$	d	f	l	m	o
a	b	b	f	l	>>	>>	>>	>>	m	o

Figure 12. The optimal alignment of σ_1 and N_3

$\gamma_5 =$

a	b	d	f	l	m	$p1$	d	f	l	m	o
a	b	b	f	l	m	$>>$	$>>$	$>>$	$>>$	$>>$	o

Figure 13. The optimal alignment of σ_2 and N_3

$\gamma_6 =$

a	b	d	f	l	$p1$	e	g	l	n	o
a	b	b	f	l	$>>$	$>>$	$>>$	$>>$	n	o

Figure 14. The optimal alignment of σ_3 and N_3

$\gamma_7 =$

a	b	d	f	l	m	$p1$	e	g	l	m	o
a	b	b	f	l	m	$>>$	$>>$	$>>$	$>>$	$>>$	o

Figure 15. The optimal alignment of σ_4 and N_3

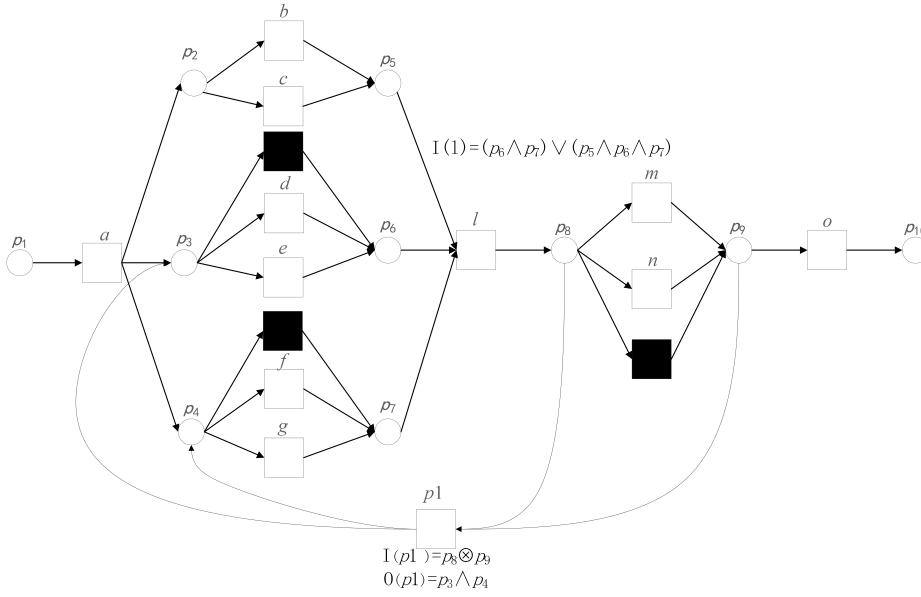


Figure 16. The model repaired with our approach

	$\lvert P\rvert$	$\lvert F\rvert$	$\lvert T+\tau\rvert$	$\lvert T\rvert$
Our approach	0	10	4	0
Fahland's method	6	24	12	8
Knapsack's method	0	22	13	12

Table 1. A comparision of added elements via the three approaches

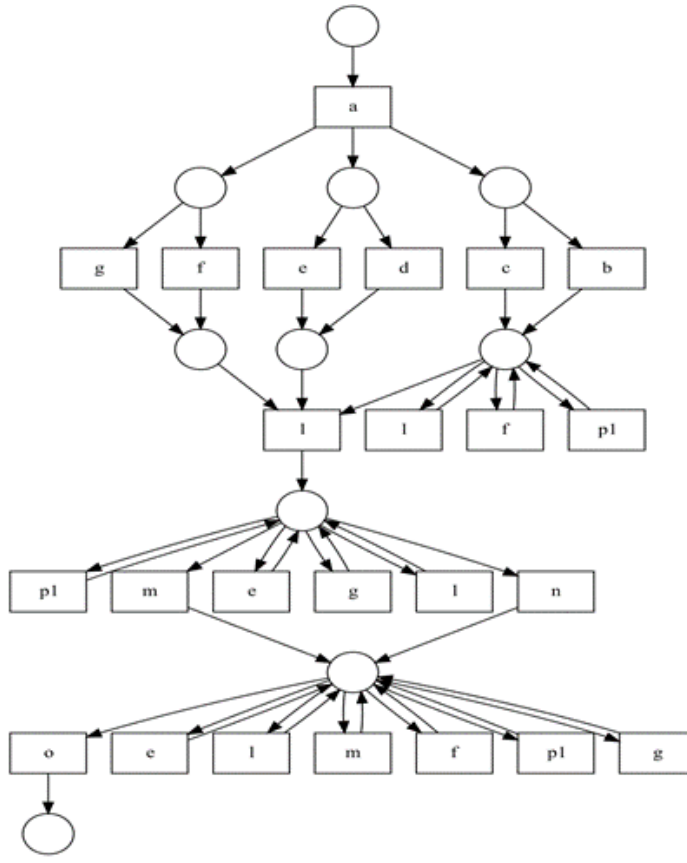


Figure 17. The model repaired with Knapsack's method [18]

4 EXPERIMENTAL EVALUATION

Some experiments are carried out in this section. Knapsack's method [18] and Fahland's method [19] are used for comparison with our proposed approach. A thoracic surgery in a hospital in Tsingtao is used to obtain the process model; `https://pan.baidu.com/s/1dH1U8SygguNwT80aMYfojg` has access to the event logs; and the Process Mining Toolkit ProM6.6 is available from `http://www.promtools.org/prom6/`, which has implemented Knapsack's method [18] and Fahland's method [19]. The method proposed in this paper is obtained by manual simulation.

4.1 Experiment Data

A business process about outpatient examination from thoracic surgery in a hospital is presented as an example. The process model of hospital outpatient examination is

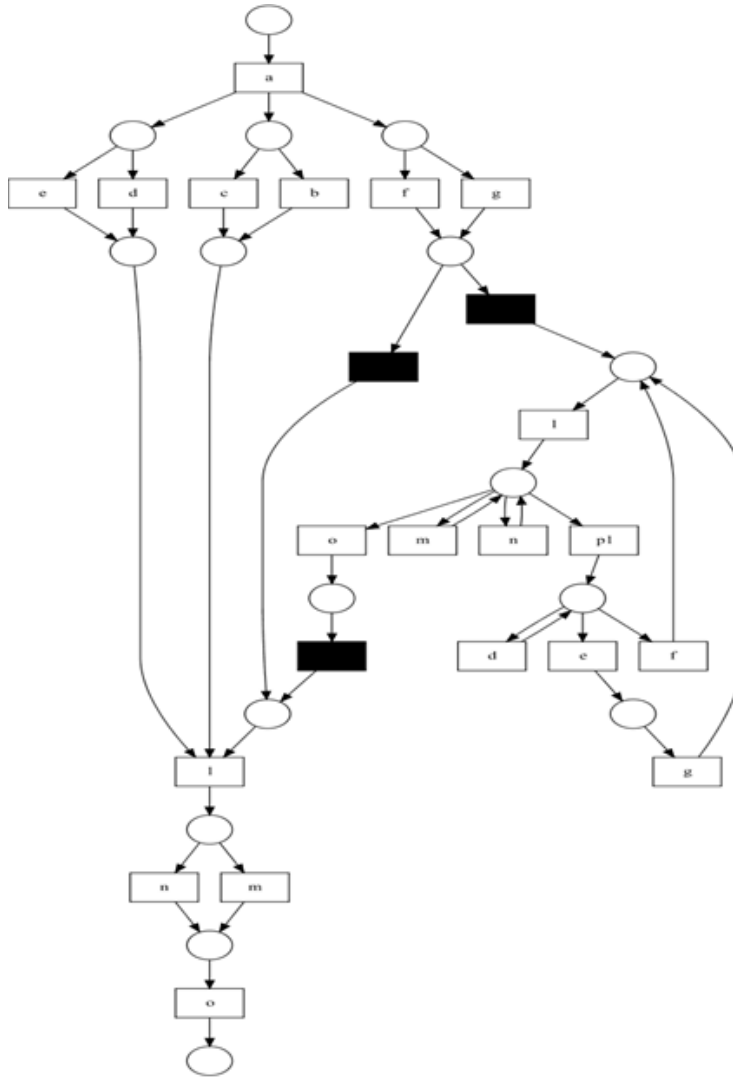


Figure 18. The model repaired with Fahland's method [19]

shown in Figure 19. The main process is described as follows: the doctor checks the patient's needs through an outpatient examination. Two types of CT examination are given as follows: conventional CT and PET-CT. Four types of clinical examination are presented as follows: gas analysis, blood routine, ESR and biochemistry. Another one is presented as follows: the contrast examination. The results of the examination are regarded as a standard to diagnose the patient's condition and the diagnosis is confirmed. Finally, the patients start the treatment.

The corresponding process is used to obtain the event logs. The traces of these event logs that deviate significantly from the process model need to be ignored first. Table 2 presents the detailed log information.

Log	Log		Total Deviation
	Trace Count	Length	
$L1$	2 056	7–10	4 523
$L2$	2 263	7–10	4 807
$L3$	2 550	7–10	5 029

Table 2. Three groups of detailed log information

4.2 Model Repair Experiments Via LPNs

The repaired models by Knapsack's method [18] and Fahland's method [19] are used as a reference to compare and analyze the effectiveness of our proposed approach in this subsection. The idea of Knapsack's method [18] is that an alignment approach is used to identify the deviations between model moves and log moves. The sub-processes are mined based on existing methods, and the sub-logs are collected based on these deviations. The original model adds a self-loop structure for repairing the sub-logs. Also, the cost of alignment needs to be considered by using the Knapsack's method. Many different repair schemes are compared by setting the cost of model moves and log moves separately. Knapsack's method can obtain the 'best' model repair approach by computing the minimum total cost of model moves and log moves. We know that the repaired model contains no invisible transitions based on Knapsack's method. Besides, the accuracy of the model will be significantly reduced based on both methods.

There are some differences in the actual process. After the patients finish the biochemical full set, they choose the blood routine or the blood gas analysis to finish. After the patients finish the biochemical full set, they choose the blood gas analysis but not the blood routine to finish. After the patients finish the blood gas analysis, the erythrocyte sedimentation rate can be chosen. Therefore, we need to repair this part. Figures 20, 21 and 22 show the repaired model by Knapsack's method [18], Fahland's method [19] and our proposed approach, respectively.

Compared with the original model, Figure 20 shows that 34 flow relations, 17 transitions, a new transition, and 16 repetitive transitions are added in the repaired model. Figure 21 shows that 12 places, 60 flow relations, and 6 invisible

transitions, 19 transitions, and a new transition, and 12 repetitive transitions are added in the repaired model. The repaired models obtained by these two methods have dramatically increased complexity. Besides, the repaired models are different from the original model. Compared with the original model, Figure 22 shows that 3 invisible transitions, 11 flow relations, and 4 transitions, but no repetitive activities are added in the repaired model. Thus, our proposed approach can be regarded as a better approach for constructing and repairing process models containing loop return structures via LPNs.

Tables 3, 4 and 5 show the experimental results of the three repair approaches, including added repetitive transition count $|T|$, added transition count $|T + \tau|$, added flow relation count $|F|$, added place count $|P|$, and the fitness and precision values, respectively. Figures 23 and 24 show the changes of the fitness and precision, respectively. An average of the three logs is used as a reference to illustrate the changes of fitness and precision. Figure 23 shows the fitness of the three approaches. As the increment of the trace count, the fitness of Knapsack's method [18] and Fahland's method [19] is not as good as that of our proposed method.

As the increment of the trace count, the precision of Knapsack's method [18] and Fahland's method [19] becomes lower and lower as shown in Figure 24. However, the precision of our approach maintains a relatively high level.

	$\|P\|$	$\|F\|$	$\|T+\tau\|$	$\|T\|$	Fitness	Precision
$L1$	0	34	17	16	0.9674	0.8123
$L2$	0	34	17	16	0.9689	0.8135
$L3$	0	34	17	16	0.9703	0.8143

Table 3. The result by Knapsack's method

	$\|P\|$	$\|F\|$	$\|T+\tau\|$	$\|T\|$	Fitness	Precision
$L1$	12	60	19	12	0.9563	0.7635
$L2$	12	60	19	12	0.9582	0.7653
$L3$	12	60	19	12	0.9603	0.7701

Table 4. The result by Fahland's method

	$\|P\|$	$\|F\|$	$\|T+\tau\|$	$\|T\|$	Fitness	Precision
$L1$	0	11	4	0	0.9825	0.9207
$L2$	0	11	4	0	0.9877	0.9232
$L3$	0	11	4	0	0.9903	0.9263

Table 5. The result by our proposed approach

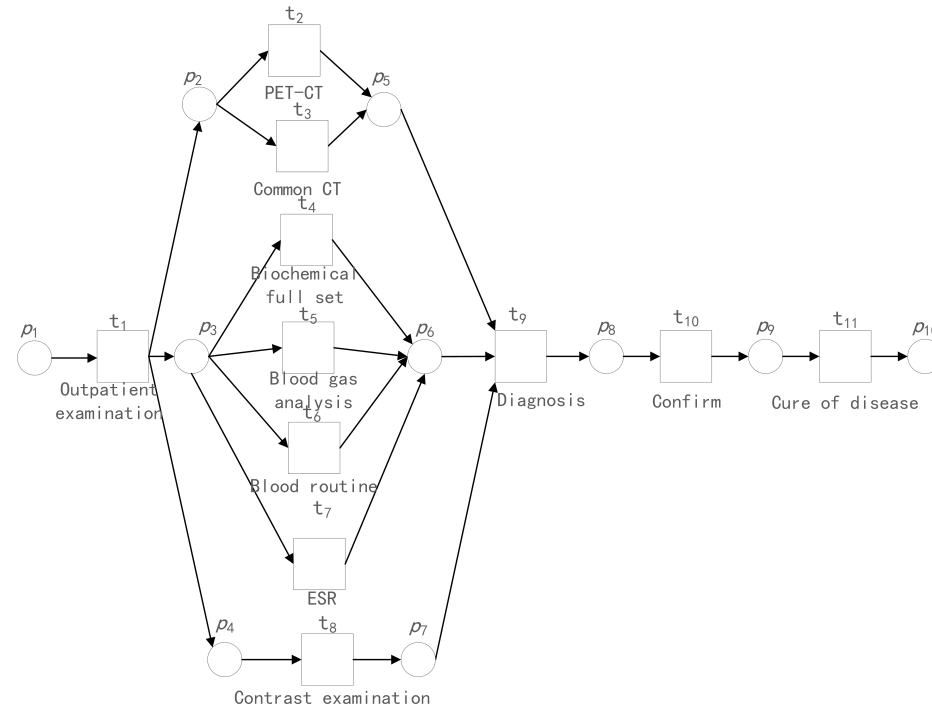


Figure 19. The process model of hospital outpatient examination

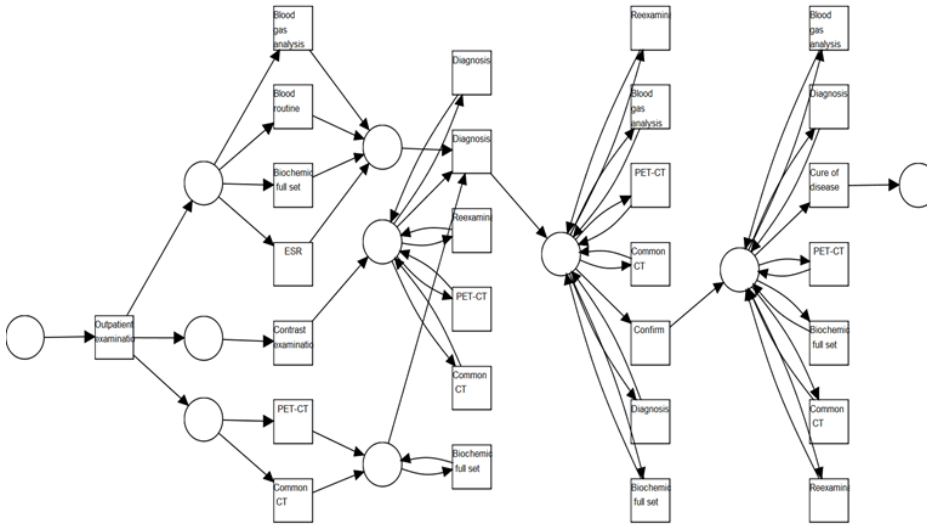


Figure 20. The model repaired with Knapsack's method [18]

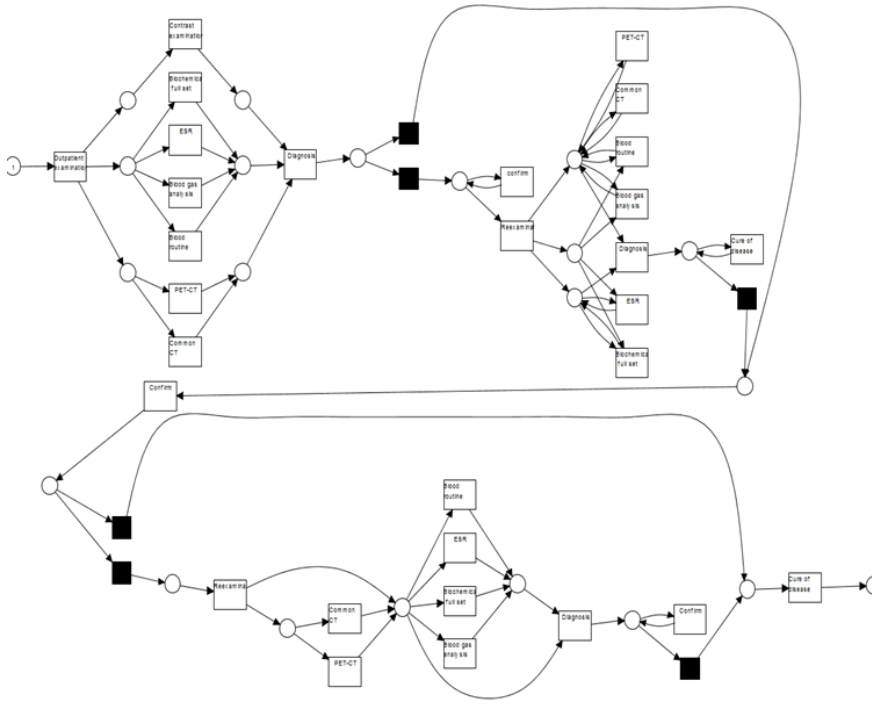


Figure 21. The model repaired with Fahland's method [19]

5 CONCLUSIONS

This paper deals with the event logs containing loop sub-traces and obtains the repaired models that can reproduce such traces. It proposes a new model repair approach based on a logic Petri net (LPN) by identifying the deviations between the models and the logs. Then, a method is designed to construct the process models containing a loop return structure. We project the activities in event logs onto the transitions in the model, and then the new activities are obtained. We get the log moves and the model moves based on the optimal alignment. Then, the relations among these activities, which are required to repair and the new activities are obtained. The new activities are regarded as the loop return transitions. We get the places that need to be controlled. Logic transitions in LPN are constructed. The logic expressions can be constructed based on these logic transitions, which are used to repair the new models. The repaired models by Fahland's method and Knapsack's method can be used as a reference to compare and analyze the effectiveness of our proposed approach. As the increment of the trace count, the fitness and precision of Fahland's method and Knapsack's method are not as good as those of our proposed approach. In this paper, we mainly consider fitness and precision as the criteria to

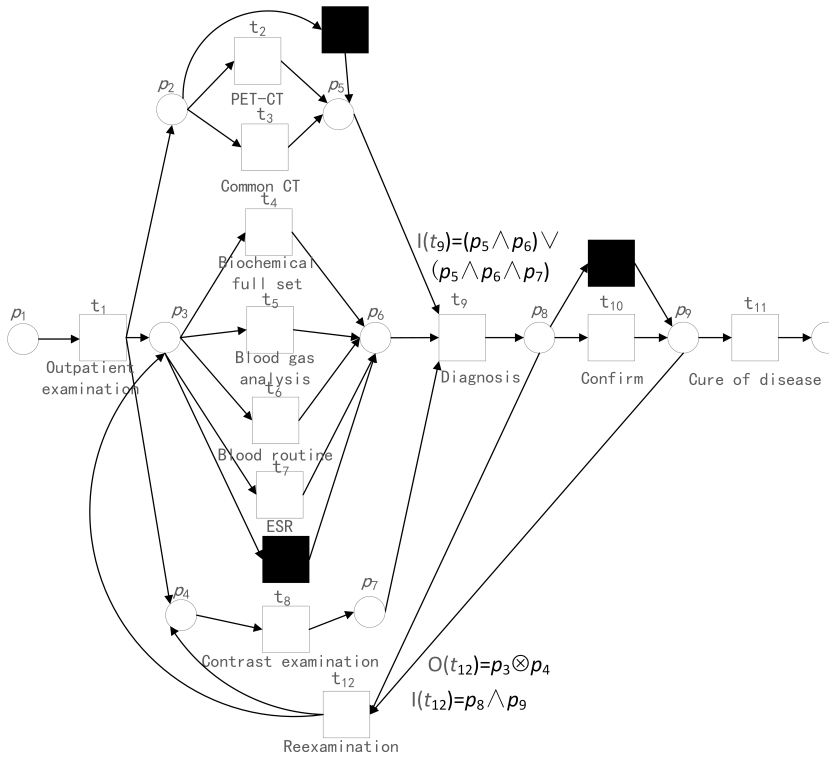


Figure 22. The model repaired with our approach

judge whether our method is suitable or not. In our future work, we will conduct a more comprehensive experimental evaluation with considering other conformance metrics such as generalization. The main concern in this paper is to obtain the process models containing loop return structures. More complex structures should

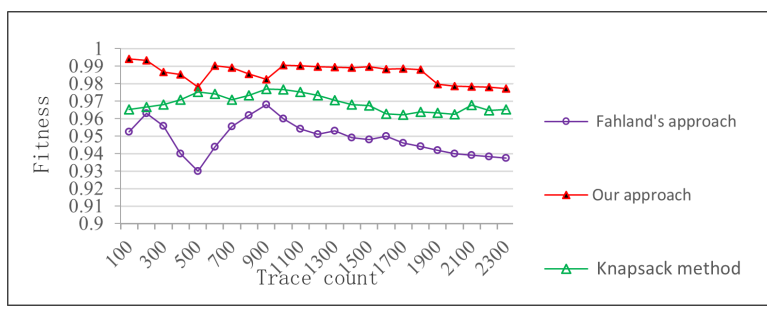


Figure 23. Fitness by experiments

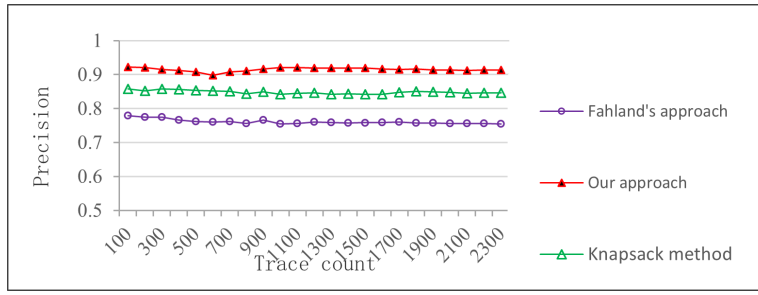


Figure 24. Precision by experiments

be considered to guarantee that a good repair effect can be obtained by our proposed approach in the future work. Some other extended Petri nets [28, 29, 30] will be found in the future for conducting process mining.

REFERENCES

[1] Sakai, M.—Takahashi, K.: Constructing a Service Process Model Based on Distributed Tracing for Conformance Checking of Microservices. NOMS 2022 – 2022 IEEE/IFIP Network Operations and Management Symposium, 2022, pp. 1–6, doi: 10.1109/NOMS54207.2022.9789804.

[2] Brzychczy, E.—Żuber, A.—van der Aalst, W.: Process Mining of Mining Processes: Analyzing Longwall Coal Excavation Using Event Data. IEEE Transactions on Systems, Man, and Cybernetics: Systems, Vol. 54, 2024, No. 5, pp. 2723–2734, doi: 10.1109/TSMC.2023.3348496.

[3] Nedopetalski, F.—de Freitas, J. C. J.: Process Mining and Simulation for a p-Time Petri Net Model with Hybrid Resources. 2021 Systems and Information Engineering Design Symposium (SIEDS), 2021, pp. 1–6, doi: 10.1109/SIEDS52267.2021.9483768.

[4] Marin, E. M. S.—Rivera, V. M. Y.—Armas-Aguirre, J.—Aguirre, S.: A Process Discovery and Conformance Checking Integration System for the Optimization of Resources in the Application of Process Mining. 2023 18th Iberian Conference on Information Systems and Technologies (CISTI), 2023, pp. 1–6, doi: 10.23919/CISTI58278.2023.10211579.

[5] Batyuk, A.—Voityshyn, V.: Streaming Process Discovery Method for Semi-Structured Business Processes. 2020 IEEE Third International Conference on Data Stream Mining & Processing (DSMP), 2020, pp. 444–448, doi: 10.1109/DSMP47368.2020.9204201.

[6] Khaosanoi, L.—Limpiyakorn, Y.: Conformance Checking and Discovery of Information Service Request Process. 2021 14th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), 2021, pp. 1–5, doi: 10.1109/CISP-BMEI53629.2021.9624359.

[7] Sitova, I.—Pecerska, J.—Merkurjevs, J.: Customer-Centric Business Process Discovery Using Process Mining Techniques. 2023 IEEE 64th International Scientific Conference on Information Technology and Management Science of Riga Technical University (ITMS), 2023, pp. 1–5, doi: 10.1109/ITMS59786.2023.10317777.

[8] Amjad, A.—Ul Haq, S.—Abbas, M.—Arif, M. H.: UML Profile for Business Process Modeling Notation. 2021 International Bhurban Conference on Applied Sciences and Technologies (IBCAST), 2021, pp. 389–394, doi: 10.1109/IBCAST51254.2021.9393223.

[9] Tsiri, T. P.—Daniyan, I. A.—Mpofu, K.: Development of a Business Process Modelling Framework for Continuous Improvements in Organisations. 2022 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), 2022, pp. 675–680, doi: 10.1109/IEEM55944.2022.9989810.

[10] Rahayu, P.—Setvowati, E.—Novianti, R.: Enterprise Modeling for Improving Company Core Busiess Using Business Process Re-Engineering. 2023 6th International Conference of Computer and Informatics Engineering (IC2IE), 2023, pp. 362–367, doi: 10.1109/IC2IE60547.2023.10331391.

[11] van der Aalst, W.—Adriansyah, A.—de Medeiros, A. K. A.—Arcieri, F.—Baier, T. et al.: Process Mining Manifesto. In: Daniel, F., Barkaoui, K., Dustdar, S. (Eds.): Business Process Management Workshops (BPM 2011). Springer, Berlin, Heidelberg, Lecture Notes in Business Information Processing, Vol. 99, 2012, pp. 169–194, doi: 10.1007/978-3-642-28108-2_19.

[12] Vij, A.—Kaswan, K. S.—Bawa, C.: Comparative Analysis of Workflow Management System Using Agents and Multi-Agents in Colored Petri Nets. 2022 IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE), 2022, pp. 1–6, doi: 10.1109/ICDCECE53908.2022.9792729.

[13] Hu, Y.—Wang, J.—Liu, G.: Resource-Oriented Timed Workflow Nets and Simulation Tool Design. 2021 International Conference on Cyber-Physical Social Intelligence (ICCSI), 2021, pp. 1–6, doi: 10.1109/ICCSI53130.2021.9736233.

[14] Yamaguchi, S.—Ahmadon, M. A. B.: A Token-Replay-Based Conformance Checking Method for Dataflow in IoT System. 2022 37th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC), 2022, pp. 1027–1030, doi: 10.1109/ITC-CSCC55581.2022.9894958.

[15] Banham, A.—Ter Hofstede, A. H. M.—Leemans, S. J. J.—Mannhardt, F.—Andrews, R.—Wynn, M. T.: Comparing Conformance Checking for Decision Mining: An Axiomatic Approach. IEEE Access, Vol. 12, 2024, pp. 60276–60298, doi: 10.1109/ACCESS.2024.3391234.

[16] Liang, J.—Yang, M.—Yang, J.—Deng, Y.: Identification of Human Operation Deviation Using a Conformance Checking Technique. 2021 International Conference on Power System Technology (POWERCON), 2021, pp. 551–555, doi: 10.1109/POWERCON53785.2021.9697743.

[17] Wen, L.—Wang, J.—van der Aalst, W. M. P.—Huang, B.—Sun, J.: Mining Process Models with Prime Invisible Tasks. Data & Knowledge Engineering, Vol. 69, 2010, No. 10, pp. 999–1021, doi: 10.1016/j.datak.2010.06.001.

[18] Polyvyanyy, A.—Van Der Aalst, W. M. P.—Ter Hofstede, A. H. M.—

Wynn, M. T.: Impact-Driven Process Model Repair. ACM Transactions on Software Engineering and Methodology (TOSEM), Vol. 25, 2016, No. 4, Art. No. 28, doi: 10.1145/2980764.

[19] Fahland, D.—van der Aalst, W. M. P.: Model Repair – Aligning Process Models to Reality. Information Systems, Vol. 47, 2015, pp. 220–243, doi: 10.1016/j.is.2013.12.007.

[20] Qi, L.—Su, Y.—Zhou, M. C.—Abusorrah, A.: A State-Equation-Based Backward Approach to a Legal Firing Sequence Existence Problem in Petri Nets. IEEE Transactions on Systems, Man, and Cybernetics: Systems, Vol. 53, 2023, No. 8, pp. 4968–4979, doi: 10.1109/TSMC.2023.3241101.

[21] Su, Y.—Qi, L.—Zhou, M. C.: A Backward Algorithm to Determine the Existence of Legal Firing Sequences in Ordinary Petri Nets. IEEE Robotics and Automation Letters, Vol. 8, 2023, No. 6, pp. 3190–3197, doi: 10.1109/LRA.2023.3246384.

[22] Luo, H.—Liu, X.—Liu, J.—Yang, Y.—Grundy, J.: Runtime Verification of Business Cloud Workflow Temporal Conformance. IEEE Transactions on Services Computing, Vol. 15, 2022, No. 2, pp. 833–846, doi: 10.1109/TSC.2019.2962666.

[23] Garcia-Uribe, C.—López-Mellado, E.: Building Hierarchical Workflow Nets for Discrete-Event Processes Discovery. 2023 20th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE), 2023, pp. 1–7, doi: 10.1109/CCE60043.2023.10332857.

[24] van der Aalst, W. M. P.—van Hee, K. M.—ter Hofstede, A. H. M.—Sidorova, N.—Verbeek, H. M. W.—Voorhoeve, M.—Wynn, M. T.: Soundness of Workflow Nets: Classification, Decidability, and Analysis. Formal Aspects of Computing, Vol. 23, 2011, No. 3, pp. 333–363, doi: 10.1007/s00165-010-0161-4.

[25] Adriansyah, A.—Munoz-Gama, J.—Carmona, J.—van Dongen, B. F.—van der Aalst, W. M. P.: Alignment Based Precision Checking. In: La Rosa, M., Soffer, P. (Eds.): Business Process Management Workshops (BPM 2012). Springer, Berlin, Heidelberg, Lecture Notes in Business Information Processing, Vol. 132, 2012, pp. 137–149, doi: 10.1007/978-3-642-36285-9_15.

[26] Wang, L.—Du, Y. Y.—Liu, W.: Aligning Observed and Modelled Behaviour Based on Workflow Decomposition. Enterprise Information Systems, Vol. 11, 2017, No. 8, pp. 1207–1227, doi: 10.1080/17517575.2016.1193633.

[27] Leemans, S. J. J.—Fahland, D.—van der Aalst, W. M. P.: Discovering Block-Structured Process Models from Event Logs Containing Infrequent Behaviour. In: Lohmann, N., Song, M., Wohed, P. (Eds.): Business Process Management Workshops (BPM 2013). Springer, Cham, Lecture Notes in Business Information Processing, Vol. 171, 2014, pp. 66–78, doi: 10.1007/978-3-319-06257-0_6.

[28] Li, H.—You, J. X.—Liu, H. C.—Tian, G.: Acquiring and Sharing Tacit Knowledge Based on Interval 2-Tuple Linguistic Assessments and Extended Fuzzy Petri Nets. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, Vol. 36, 2018, No. 1, pp. 43–65, doi: 10.1142/S0218488518500034.

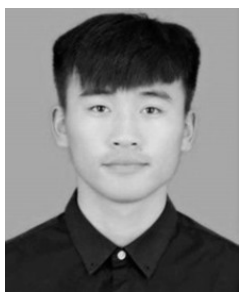
[29] Liu, H. C.—Luan, X.—Li, Z. W.—Wu, J.: Linguistic Petri Nets Based on Cloud Model Theory for Knowledge Representation and Reasoning. IEEE Transactions on Knowledge and Data Engineering, Vol. 30, 2018, No. 4, pp. 717–728, doi:

10.1109/TKDE.2017.2778256.

[30] Liu, H. C.—You, J. X.—Li, Z. W.—Tian, G.: Fuzzy Petri Nets for Knowledge Representation and Reasoning: A Literature Review. Engineering Applications of Artificial Intelligence, Vol. 60, 2017, pp. 45–56, doi: 10.1016/j.engappai.2017.01.012.

Yuyue Du received his B.Sc. degree from the Shandong University, Jinan, China, in 1982, his M.Sc. degree from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 1991, and his Ph.D. degree in computer applications from the Tongji University, Shanghai, China, in 2003. He is currently Professor at the College of Computer, Shandong Xiehe University, Jinan, China, and Professor of computer science and technology at the Shandong University of Science and Technology, Qingdao, China. He has taken in over 10 projects supported by the National Nature Science Foundation, the National Key Basic Research Developing Program, and other important and key projects at the provincial levels. He has published over 200 papers in domestic and international academic publications. His research interests are in formal engineering, Petri nets, real-time systems, Web services, and workflows.

Wenjing Luan received her Ph.D. degree in computer software and theory from the Tongji University, Shanghai, China in 2018. She is currently a Lecturer of computer science and technology at the Shandong University of Science and Technology, Qingdao, China. From May to July, 2017, she was a Visiting Scholar in the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, USA. Her current research interests include machine learning, location-based social networks, and recommender systems. She received the Best Student Paper Award-Finalist in the 13th IEEE International Conference on Networking, Sensing and Control (ICNSC 2016).



Xize Zhang received his B.Sc. degree from the Shandong University of Science and Technology, Qingdao, China, in 2016. He is now pursuing his M.Sc. degree in the College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao, China. His current research interests are process mining, Petri nets and workflow.

Mei Chen received her B.Sc. degree in computer science and technology from the Dezhou University, Dezhou, China in 2005 and her M.Sc. degree in computer application technology from the Suzhou University, Suzhou, China in 2008. She is Associate Professor at the College of Computer, Shandong Xiehe University, Jinan, China. She has published over 20 papers in domestic and international journals and conferences, including 5 SCI and EI indexed papers. Her current research interests include digital image processing and object detection.