IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 22, 2020, accepted December 29, 2020, date of publication December 31, 2020,
date of current version February 10, 2021.

Digital Object Identifier 10.1109/ACCESS.2020.3048486

Real-Time Repair of Business Processes Based on Alternative Operations in Case of Uncertainty

LIWEN ZHANG, XIANWEN FANG, CHIFENG SHAO, AND LILI WANG
College of Mathematics and Big Data, Anhui University of Science and Technology, Huainan 232001, China

Corresponding author: Xianwen Fang (1758678508@qq.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61402011, Grant 61572035, and Grant 61272153; in part by the Natural Science Foundation of Educational Government of Anhui Province of China under Grant KJ2016A208; in part by the Anhui Provincial Natural Science Foundation under Grant 1508085MF11; in part by the Academic and Technology Leader Foundation of Anhui Province, Anhui Province University Discipline (Professional), Top-notch Talent Academic Project under Grant gxbjZD11; in part by the Big Data Industry Base Project in Anhui Province; and in part by the Open Project Program of the Key Laboratory of Embedded System and Service Computing of Ministry of Education under Grant ESSCKF2018-04.

ABSTRACT Owing to the continual evolution of business processes, differences often occur between observable behavior in the event log and the actual operation of the given process model. Whether an iterative and observable deviation occurs between the event log and the process model is in general uncertain. Existing repair techniques add only observable deviations in a fixed manner, which makes it difficult to consider the fitness and the precision of the results simultaneously. To solve this limitation, this study proposes a method of repair that can improve precision as much as possible without affecting the fitness of the results as well as the number of repair-related activities. Patterns of behavior that cannot be replayed are determined by refined reachable activity structures and conformance checks of behavioral relationships. They are optimized by constructing repair or configuration operations. The two operations can be switched based on the identification of iterative observable deviation in the given pattern of behavior during repair. To assess this, datasets from simulated and empirically acquired business processes were used. The proposed method improved the precision of six datasets by 12% on average. Deviations were repeatedly generated by the cycle, because of which the average precision improved by 21% on three datasets with loops.

INDEX TERMS Refined reachable activity structure, optimal alignment, configured replay graph, repair operation, configuration operation.

I. INTRODUCTION

A. RESEARCH BACKGROUND

Process mining involves using knowledge extracted from event logs to formulate a process model, and improving its performance by using a variety of technologies and tools [1]. A business process can be analyzed through process mining from three perspectives: process discovery, conformance check, and process enhancement [2]. Process discovery and the conformance check are the most important parts of process mining [3], [4]. Model repair is a recently developed type of process mining between process discovery and the conformance check that considers behaviors that cannot be replayed on the process model, and forms a repaired process model that is as similar as possible to the original model [5].

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

Model repair is executed according to the differences detected by the conformance check [6]. The conformance check can be applied to a variety of settings, including compliance auditing, model maintenance, and automated process discovery. The scenario in which the event log cannot be replayed on the process model can be identified by a compliance audit, i.e., as non-replayable behavior [7]. Model maintenance can be used to diagnose not only non-replayable behaviors but also behaviors unique to the given model [8]. Automatic process discovery is the adjustment of the process model according to the output of the conformance check. The initial model is automatically adjusted by removing behaviors unique to the model or adding behaviors unique to the event log. To reduce the elements of deviation between the event log and the process model to as few as possible, the A^* search method can be used to construct the optimal alignment [9]. The deviations with the same label and location are regarded as one repair

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

activity. It is worth noting that in order to calculate the fitness and precision accurately, occurrence of each deviation needs to be calculated as an independent cost (this includes the iterative occurrence of deviation) [10].

The criteria used to assess model repair include fitness, precision, simplicity, and generalization, which are independent and affect each other. Fitness refers to the degree of replay of the event log in the process model, and precision is the accuracy of the replay. Simplicity determines the structural complexity of the repaired model while generalization represents the similarity between the repaired model and the initial model [11]. The fitness and precision are most important for assessing the performance of model repair. The event log should not only be completely replayed to the process model by repair but the accuracy of replay should be improved as much as possible (i.e., if an activity occurs independently or iteratively in the event log, the corresponding activity in the process model should also occur in a consistent manner). Model repair deals with deviation between the event log and process model during replay, which includes two types [12]: i) the observable deviation only occurred in the event log. The repair performance of this deviation involves fitness and precision. ii) The skip deviation produced by the process model only affects fitness [13]. The existing repair methods mainly include the following two in terms of precision: i) each observable deviation is inserted on the process model using self-loop, which repairs all iterative observable deviations accurately [14], [15]. This technology can easily lead to under-fitting when there are few iterative deviations in the event log, i.e., precision is too low [16]. Precision is improved to a certain extent by building into a concurrent substructure the observable deviations in the event log that follow directly[1] [17]. ii) The conflict substructure constructed by the invisible transition and the observable deviation is added to the process model. The precision after repair is one when the iterative activity is not contained in the initial process model [13]. However, this comes at the cost of fitness when iterative deviation is produced by the self-loop in the event log. Fig.1 (a) and (b) describe the two repairs of the process model w.r.t the event log $L = ABBBCDE$, respectively. B, B^1, B^2 and D are only observed in the event log. B^1 and B^2 are produced by the iteration of B. All deviations are replayed, whereas B and D are not accurately replayed in Fig.1 (a). The fitness after repair is one, but precision is significantly affected in this case. As shown in Fig.1 (b), L needs to be preprocessed as $L' = ABCDE$ for measure of precision. The iterative activity produced by self-loop is not contained in initial process model in Fig.1 (b), so the precision after repair is one. B^1 and B^2 cannot be replayed, so the fitness after repair is less than one in Fig.1 (b).

[1]The directly following observable deviations are adjacent deviations in the event log, which can be repaired at the same place of model.

The substructure is used here according to a single-in and single-out network structure, which is included in the process model.

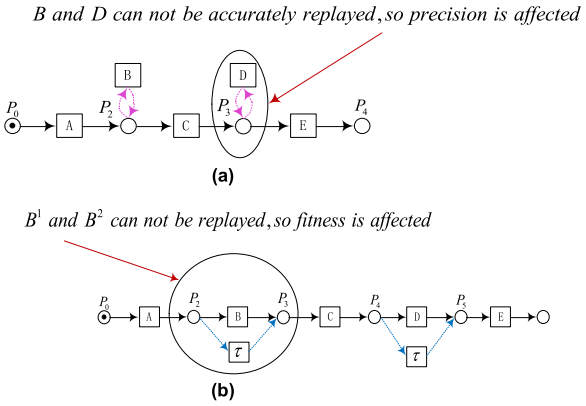

FIGURE 1. Two repairs of process model w.r.t *L*.

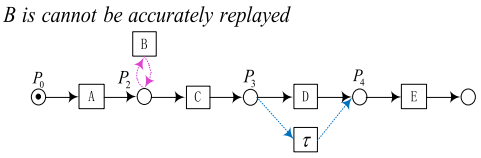

FIGURE 2. The improved repair of process model w.r.t *L*.

B. SOLUTION AND CONTRIBUTIONS

With the aim of solving the above problems, the new repair method is proposed in this paper. The appropriate operation is chosen to repair the following two behavioral patterns that cannot be replayed [18]. 1) The behavioral pattern containing iterative observable deviation. Although the non-iterative observable deviation in this behavior pattern cannot be accurately repaired, all observable deviations can be repaired using the self-loop inserts. 2) The behavioral pattern without the iterative observable deviation. The principle of configuration optimization is introduced to the model repair technology in this case. The customized business process is initiated by a subset of the reference process model, and such processes are varied according to the requirements of stakeholders under the given constraints [19]. This paper proposes that common and variable parts in a configurable process are behaviors that can and cannot be replayed, respectively, between the event log and the process model. They are also respectively called fitted and unfitted behaviors. Configuration adds observable deviation on the process model by the conflict substructure that consists of the invisible transition and this deviation. All observable deviations in the behavioral pattern without iterative observable deviation can be accurately repaired by configuration. This paper focuses on improving precision as much as possible while ensuring that the fitness is 1 and without increasing the number of repair activities (the simplicity and generalization are not affected). Fig.2 is the improved repair of Fig.1 (a) and (b). The noise problem is beyond the scope of this article, and the event log is considered as a set of filtered sequences of execution [20].

The remainder of this article is structured as follows: The preliminaries required for real-time repair are described in Section 2. The differences between the event log and the process model are identified from two perspectives in Section 3, and real-time repairs to various unfitted behavioral patterns are introduced in Section 4. The feasibility and effectiveness of the proposed method are verified in Section 5 through experimental evaluation, and Sections 6 and 7 respectively describe related work in the area and the conclusions of this study.

II. PRELIMINARIES

Definition 1 (Event Log): The event log is a six-tuple $L = (E, C, \mu, \ell, o, \aleph, \propto)$, where E represents all events in the log, $\forall e \in L$. c is uniform identifier for a trace that occurs multiple times during system operation, $\forall c \in C$. μ associates each event in a trace to a case, $\mu(e_i) = c$. ℓ assigns a label to corresponding an event, $\ell(e) = a, \forall a \in \aleph$. o converts a label into an event, $o(a) = \ell(e)$. $\aleph$ is the label set of all events, and $\ell(e) \in \aleph, \forall e \in L$. $\propto$ represents the weak order relationships between adjacent events in the log, $\propto \subseteq E \times E$.

Definition 2 (Labeled WF-Net System): A tuple $N = (P, T, F, Z, \lambda, \sigma, M_i)$ is set as the labeled Petri net system, where P is a finite set of places, T is a finite set of transitions, F represents the flow relationships between adjacent places and transitions,$F \subseteq (P \prec T) \cup (T \prec P)$. Z is a finite set of labels in the Petri net system,$Z = Z \cup \{\tau\}$. λ assigns labels to the corresponding transitions in the Petri net system, $\lambda : T \leftarrow Z \cup \{\tau\}$, and $\forall \lambda(t) \in Z \cup \{\tau\}$. σ converts a label into a transition, $\sigma(a) = \lambda(t)$. M_i is the initial marking in net system N. This paper restricts the Petri net system to a safe one, and thus $M_i(p) = 1$. The labeled Petri net system can be regarded as a labeled WF-net system $N = (P, T, F, Z, \lambda, \sigma, M_i, p_i, p_f)$ if and only if $|p_i| = 1 \wedge |p_f| = 1$, where $p_i \in P$ and $p_f \in P$ represent the initial and the final places in the net system, respectively.

Definition 3 (Unfitted Behavioral Pattern): Behavioral pattern consists of several activities in event log or process model and behavior relationship between them. Model repair exclusively considers differences produced when the event log cannot be replayed on the process model. The unfitted behavior is caused by the corresponding patterns between the event log and the process model during replay. Each pair of the corresponding patterns that causes unfitted behavior can be merged into an unfitted behavioral pattern based on the conformance check of behavioral relationships. $B_{(L/N^*)(+\setminus\times\setminus\prec)}$ is the behavioral pattern produced by unfitted behavioral relationships (concurrency/causality/conflict). $B_{(L/N^*)\subseteq}$ represents the behavioral inclusion pattern, which contains behavior that occurs only in the event log or the process model. $B_{\ell(e)}$ and $B_{\lambda(t)}$ represent an activity that can be observed in the event log but cannot be captured in the process model, and an activity in the process model that prevents the event log from being replayed, respectively.

Definition 4 (Optimal Alignment): Alignment is a set of comparisons between the executive sequences in the event log and the corresponding firing sequences in the process model, and is denoted by ξ^*. Let M_{LN^*} and M_L/M_{N^*} be sets of synchronous and asynchronous moves produced by the alignment, respectively. The deviation results from either M_L or M_{N^*}; thus, it is divided into the following two types: i) the insert deviation that can be observed on the event log but cannot be captured in the process model is denoted by $Esert(\ell(e)), \forall e \in L$; ii) the skip deviation that belongs only to the process model and causes the event log to not be replayable on the process model, where this is denoted by $Skip(\lambda(t)), \forall t \in T$. The distance between the corresponding activities in the set of asynchronous moves is set to one, and otherwise to zero: $Dis(M_L) = Dis(M_{N^*}) = 1$ and $Dis(M_{LN^*}) = 0$. Optimal alignment minimizes the distance between the event log and the process model by comparing the corresponding sequences, $Dis(\xi_{op}) \leq Dis(\xi), \xi_{op} \in \xi^*$, and $\forall \xi \in \xi^*$.

Definition 5 (Configured Replay Graph): The configured replay graph is a four-tuple $C_G = (C, T, Add\Gamma, Hide\Gamma)$. C represents the set of execution configurations of reachable activities during replay. Because the execution configuration of any activity is the set of several sequences of reaching this activity from an initial activity, the set of execution configurations in the reachable activity structure is $C(\wp)$. Based on the nesting performance of the cycle, it can be inferred that there is no the maximized size of the set of execution configurations in the reachable activity structure in cycle. Γ is the set of labels that connect two adjacent execution configurations, $Add\Gamma$ is the label set of activities that can be observed on the event log but cannot be captured in the net system, $\forall Add(\ell(e)) \in Add\Gamma$. $Hide\Gamma$ is the label set of activities in the net system that prevent the event log from being replayed properly, $\forall Hide(\lambda(t)) \in Hide\Gamma$.

Definition 6 (Iterative Insert Deviation): The iterative insert deviation is a special deviation produced by the self-loop in the event log. The iterative insert deviation includes the following two types: i) the insert deviation that is produced only by the self-loop, $IEsert\ell(e)^n$; ii) the same insert deviation that is produced by the both the self-loop and the unfitted behavior, $IIEsert\ell(e)^n$. n denotes the number of iterations of the iterative insert deviation. The non-iterative deviation produced by unfitted behavior is included in $IIEsert\ell(e)^n$. Thus, the number of iterations of $IIEsert\ell(e)^n$ is counted from the second occurrence, whereas $IEsert\ell(e)^n$ needs to record all occurrences. The iterative insert deviation may occur once or multiple times in different cases of the event log and the cost per iteration is one.

Definition 7 (The Repair Operation of Unfitted Behavior): The repair operation detects the complete information of deviation by the optimal alignment and repairs the process model using the self-loop insert or invisible transition skip, R_O. According to the different types of deviations, the repair operation can be divided into the following two parts: i) the insert/skip deviation can be repaired by the self-loop or invisible transition, $R_O(a)$; ii) the directly following deviations that satisfy a certain behavioral relationship can be constructed

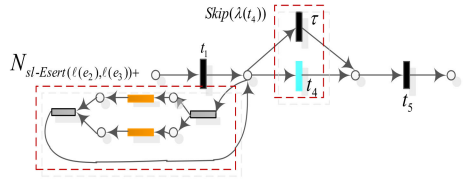

FIGURE 3. An example of the repair operation.

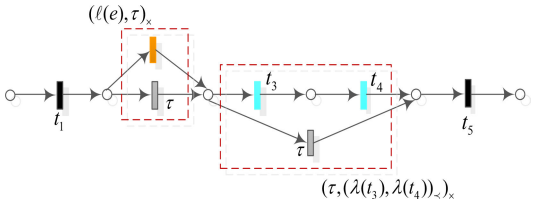

FIGURE 4. An example of the configuration operation.

TABLE 1. Symbols of definition 5-8.

Equation no.	Usage
$C(\wp)$	$\sum_{a\in\Gamma} c(a) = \sum_{i=1}^{\|\Gamma\|}(a_0,\ldots,a_i)$
$IEsert1(e)^n$	$\|IEsert1(e)^1,\ldots,IEsert1(e)^n\|$
$IEEsert1(e)^n$	$\|Esert1(e)\cup(IEsert1(e)^1,\ldots,IEsert1(e)^{n-1})\|$
$R_O(a)$	$sl-Esert1(e)/(\tau,\lambda(t))_\times$
$R_O(s)$	$N_{sl-E(1(e_1),1(e_2))+/p/\times} / N_{S(\lambda(t_1),\lambda(t_2))+/p/\times}$
$C_O(a)$	$(1(e),\tau)_\times/(\tau,\lambda(t))_\times$
$C_O(s)$	$((1(e_1),1(e_2))_{+/p/\times}/(\tau,(\lambda(t_1),\lambda(t_2))_{+/p/\times})_\times$

into substructure and repaired by the self-loop insert/skip, $R_O(s)$. Fig.3 describes the repair operations of the inserted concurrent substructure and skipped deviation.

Definition 8 (The Configuration Operation of Unfitted Behavior): Configuration optimization analyzes all mismatched behaviors between business processes. However, this paper considers only the unfitted behavior in the replay, $C(L\diamond N^*)_{uf}$. The configuration operation adds or removes the conflict substructure formed by the configurable behavior and invisible transition on the process model, C_O [21]. Configuration operations can be divided into the following two types according to configurable behavior: i) the configurable activity is added or hided on process model by invisible transition, $C_O(a)$; ii) the configurable substructure is added or hided on process model by invisible transition, $C_O(s)$. Fig.4 describes the configuration operations of added activity and removed causal substructure (skip and remove are consistent).

Some of the symbols in this section and their usages are recorded in the table 1.

III. DIAGNOSIS OF THE PROBLEM

The event log and the process model may have exhibit unexpected and inconsistent behaviors during operation. The event log cannot be replayed on the process model, and leads to unfitted behavior. This problem needs to be solved by model repair. This section introduces two methods for the diagnosis of differences in unfitted behavioral patterns from different perspectives.

A. PARTITION OF BEHAVIORAL PATTERNS

The refined reachable activity structure is improved on the basis of the refined process tree [22]. Fragments of the reachable activity structure are divided layer by layer from whole to part according to different behavior relations. When the reachable activity structure is refined, all corresponding fragments that satisfy the shortest distance can be discovered under the given constraints.

Definition 9 (The Refined Reachable Activity Structure): $R(\wp) = (conc, seq, conf, C(R(\wp)))$ refines the reachable activity structure into several fragments according to different behavioral relationships, i.e., concurrency, sequence, and conflict (a single activity that cannot be divided by these three behavioral relationships can be regarded as an independent behavioral pattern). *conc* represents the concurrency fragment, in which the activities occur simultaneously in any order, *seq* represents the sequence fragment, in which the behavioral relationship between the immediately following activities is causality, and *conf* represents the conflict fragment, where activities on different branches are mutually exclusive. The construction of the refined reachable activity structure during replay needs to satisfy certain constraints, that is, $C(R(\wp))$. In order of priority, the rules are as follows:

$$
\begin{aligned}
&p(R(\wp)) = (a_i, seq(\wp)/conc(\wp)/conf(\wp), a_f) \Rightarrow \\
\text{I)}\quad &(\ell(e_0) = a_i/\lambda(t_0) = a_i) \wedge (\ell(e_n) = a_f/\lambda(t_m) = a_f) \wedge \\
&|E| = n \wedge |T| = m \\
\text{II)}\quad &entry(seq/conc/conf) \prec (seq/conc/conf) \prec exit(seq/conc/conf) \\
\text{III)}\quad &|entry(seq/conc/conf)| = |exit(seq/conc/conf)| \\
\text{V)}\quad &(\textstyle\sum_{i=1}^{|seq|} a_i = S \Rightarrow |S| > 1)/ \\
&(\textstyle\sum_{i=1}^{|conc|} a_i = P \Rightarrow |P| > 1)/ \\
&(\textstyle\sum_{i=1}^{|conf} a_i = E \Rightarrow |E| > 1)/ \\
&(\textstyle\sum_{i=1}^{|\ell(E)/\lambda(T)|} a_i = A \Rightarrow |A| > 1) \\
\text{VI)}\quad &C(R(\wp_{N^*})) \approx C(R(\wp_L)) \\
\text{VII)}\quad &pre(seq) > pre(conc, conf)
\end{aligned}
$$

$pre(seq) > pre(conc, conf)$ represents the preferential division of the sequence fragment, and $C(R(\wp_{N^*})) \approx C(R(\wp_L))$ shows that refining the reachable activity structure of the process model under this constraint must minimize the distances between its behavioral patterns and the corresponding behavioral patterns in the event log [23].

Fig. 5 (a) and (b) shows the refined reachable activity structures of the event log L_1 and the network system N^*, respectively. The dashed orange, green, and blue lines are used to divide the concurrency fragment, sequence fragment, and conflict fragment, respectively, while the dashed purple line represents the initial activity and the final activity of the reachable activity structure.

According to the structures above, all corresponding fragments that satisfy the shortest distance can be obtained, and are recorded as follows:

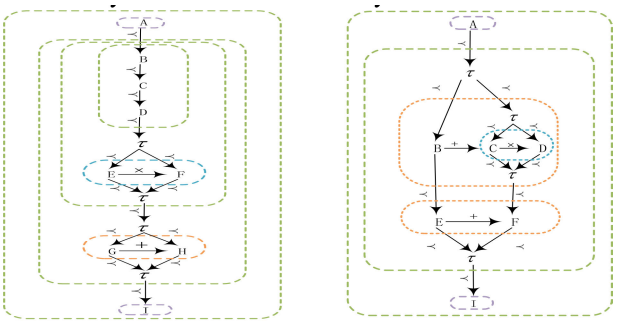

FIGURE 5. The Refine reachable active structure: (a) The event log. (b) The net system.

L_1	$\{(ABCDEGHI),(ABCDFHGI)\}$				
N^*	$\{(ABEI),(ACFI),(ADFI)\}$				
$B(\wp_{L_1})$	A	$seq(BCD)$	$conf(EF)$	$conc(GH)$	I
$B(\wp_{N^*})$	A	$conc(Bconf(CD))$	$conc(EF)$	$\varnothing$	I

The resulting corresponding fitted and unfitted behavioral patterns are:

B_{L_1}	A	$B_{L_1\prec}(BCD)$	$B_{L_1\times}(EF)$	$B_{L_1+}(GH)$	I
B_{N^*}	A	$B_{N^*+}(BB_{N^*\times}(CD))$	$B_{N^*+}(EF)$	$\varnothing$	I

B. COLLECTING COMPLETE INFORMATION ON THE DEVIATION

This section describes the following: i) the identification of iterative insert deviation, ii) the detections of occurrence and location of non-iterative insert deviation, and iii) the observation of behavioral relationship between several non-iterative insert deviations. These three kinds of deviation information are served to the repair operation in definition 7.

1) IDENTIFYING ITERATIVE INSERT DEVIATION

The iterative insert deviation is a special type of deviation, meaning that it occurs if and only if the behavior in the event log passes through a self-path of the loop.

The corresponding behavioral patterns between the initial network system N^* in Fig. 6 and the event log L_2 are merged based on the conformance check of the behavioral relationship. The consistent behaviors between the corresponding patterns are directly merged into the fitted behavioral patterns. The fitted and unfitted behavioral patterns are recorded as follows:

L_2	$\{(ABBBCCDDG)^{15},(ABBDDCCG)^{21},(ABCDG)^{20},(ABDCG)^{17}\}$		
B_{L_2/N^*}	A	$B_{N^*\subset}(B_{\prec/\prec}(BB_{+/\prec}(CD)B_{N^*\prec}(EF))$	G

The conformance check based on the behavioral relationship can detect the unique behavior on network system and the unfitted behaviors (the unique behavior on network system is $B_{N^*\prec}(EF)$). $B_{N^*\prec}(EF)$ is directly retained without further analysis, and the behavioral patterns used for replay are recorded as:

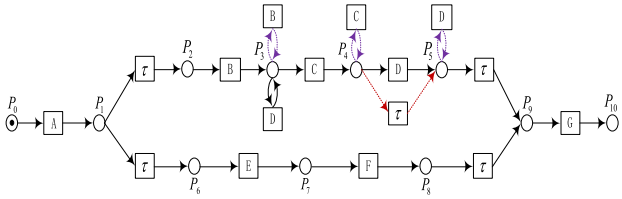

FIGURE 6. Repaired net system with iterative insert deviations.

B_{L_2/N^*}	A	B	$B_{+/\prec}(CD)$	G

The event log is completely replayed on the process model based on above behavioral patterns, resulting in the following the optimal alignment [24]:

ξ_{op1}	A	B	B	B	C	C	D	D	G
	A	B	→	→	C	→	D	→	G

ξ_{op2}	A	B	B	D	D	C	C	→	G
	A	B	→	→	→	C	→	D	G

ξ_{op3}	A	B	D	C	→	G			
	A	B	→	C	D	G			

The red letter in the optimal alignment above represents $IEsert\ell(e)^1$. Even if $IEsert\ell(e)^1$ occurs only once, it is considered an iterative insert deviation: for example, $IEsertC^1$, and $IEsertD^1$ in ξ_{op1} and $IEsertB^1$ and $IEsertC^1$ in ξ_{op2}. The purple dotted line and solid red line in Fig. 6 show the repair of these iterative insert deviations and *SkipD*, respectively. The consistency measurements between the repaired N^* and L_2 are $f = 0.91$ and $p = 0.85$, respectively. The type of deviation of D in ξ_{op2} is $IEEsertD^2 = EsertD \cup IEsertD^1$. In terms of precision, the repair of the black solid line in Fig. 6 cannot accurately replay the non-iterative insert deviation $EsertD$ in $IEEsertD^2$. The consistency measurements between the repaired N^* and L_2 are $f = 1$ and $p = 0.90$, respectively.

2) DETECTING NON-ITERATIVE INSERT DEVIATION

The occurrence and location of non-iterative insert deviation can be determined by optimal alignment. Because the optimal alignment is produced by fitted and unfitted behavioral patterns, the behavioral relationship between the directly following deviations can be discovered. Several directly following deviations are constructed into a substructure according to the detection-related information in order to reduce the size of the event log by as much as possible.

To detect complete information on non-iterative insert deviations, all corresponding behavioral patterns produced by the refined reachable activity structures are set as follows:

$B(\wp_L)$	A	$B_+(BC)$	$B_\times(DEF)$	$\varnothing$	I
$B(\wp_{N^*})$	A	$B_\times(BC)$	$\varnothing$	$B_+(GH)$	I

The fitted and unfitted behavioral patterns between the event log and the net system are recorded as follows:

B_{L/N^*}	A	$B_{+/\prec}(BC)$	$B_{L\subseteq\prec}(DEF)$	$B_{N^*\subseteq+}(GH)$	I

The following optimal alignment can be obtained based on the behavior relation that cannot be replayed (the iterative insert deviations are contained in the event log L) [25]:

ξ_{op1}	A	B	B	B	C	D	D	D	E	F	→	→	I
	A	B	→	→	C	→	→	→	→	→	G	H	I

ξ_{op2}	A	C	B	B	C	D	E	F	→	→	I		
	A	→	B	→	C	→	→	→	G	H	I		

The background colors are used above to distinguish between iterative insert deviations and non-iterative insert deviations produced by different unfitted behavioral patterns. Thus, the deviations detected by optimal alignment can be divided into the following five groups: $IEsertB^2$, $IEsertB^1$, $EsertC$, $(IEEsertD^3, EsertE, EsertF)$, and $(SkipG, SkipH)$. The following two methods are used to determine their repair locations: i) $\ell(e) \in M_L$. The location is the rear place of transition in the last synchronous movement before the asynchronous movement that causes this deviation, $loc(Esert\ell(e_i)) = (\sigma(a_{i-1}))^\bullet \wedge \ell(e_i) = a_i$. ii) $a \in M_{N^*}$. The pre and rear places of the transition in the asynchronous movement produced by this deviation are used as the start and the end of the invisible transition, respectively, that is, $loc(Skip\lambda(t)) = {}^\bullet(\sigma(\lambda(t))) \cup (\sigma(\lambda(t)))^\bullet$. The two groups of directly following non-iterative insert deviations can be extracted from the above optimal alignment: $M'_L = \{D, E, F\}$ and $M'_{N^*} = \{G, H\}$. The two substructures are constructed based on different behavioral relationships between the deviations: $N_{sl-E(D,E,F)\prec}$ and $N_{S(G,H)+}$.

C. DISCOVERING CONFIGURABLE BEHAVIOR

Configuration is an optimization technique that renders compatible mismatching behavior between the reference process model and the customized business process. The event log that does not contain the iterative insert deviation can be considered a reference process model during the discovery of configurable behavior [26]. This section introduces the configured replay graph based on the fitted and unfitted behavioral patterns produced by the event log replay on the process model. The configurable behavior between the event log and the process model can be discovered by the configured replay graph, and is served to the configuration operation in definition 8.

Fig. 7 constructs the configured replay graph based on the fitted and unfitted behavioral patterns produced by Fig.5. $B_{\prec/+}(BB_{\prec/\times}(CD))$ is replayed as $B \prec C \prec Add(D)$ to identify a configurable activity $Add(D)$. $B_{\times/+}(EF)$ is replayed as $(E \prec Hide(F))/(F \prec Hide(E))$, where the two configurable activities are exclusive and thus are used to form configurable substructures denoted by $N_{A(FE)\times}$.

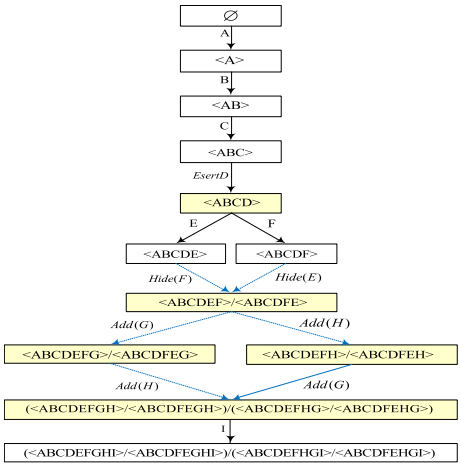

FIGURE 7. Configuration replay graph.

IV. REAL-TIME REPAIR

To improve the consistency of the business process as much as possible, the process model repair with respect to the event log needs to improve in precision and maintain its fitness. It is difficult to balance fitness and precision using the available methods. This problem is solved by the real-time repair proposed in this section.

A. PRINCIPLE AND EFFECT OF REAL-TIME REPAIR

1) LIMITATIONS OF REPAIR

The insert deviation produced by the self-loop in the event log occur an infinite number of times. Otherwise, it is occurred independently. All of deviations in a dataset can be replayed does F equal one, i.e., $C\cos t = 0$. However, only if all of $\Delta|\ell(e)|$ and $\Delta|\lambda(\sigma(\ell(e)))|$ are both 1 or ∞ does P equal one. The repair for the skip deviation does not affect precision. This deviation is removed by the invisible transition in any case. Therefore, existing methods can be divided into the following two types in terms of the repair of insert deviation: i) the insert deviation is repaired using the self-loop, i.e., $R_o(D)$. The corresponding activity of this deviation is iterative in the process model $(\Delta|\lambda(\sigma(\ell(e)))| = \infty)$. All insert deviations in a dataset can be replayed, but the non-iterative insert deviation cannot be accurately replayed. That is, $F(R_O(D)) = 1$ and $P(R_O(D)) \leq 1$ (φ). ii)The insert deviation is configured by adding $conf_c$, i.e., $C_O(D)$. The corresponding activity of this deviation is non-iterative in the process model $(\Delta|\lambda(\sigma(\ell(e)))| = \infty)$. The iterative insert deviation cannot be replayed, because of which fitness is affected to a certain extent. That is, $P(C_O(D)) = 1$ and $F(C_O(D)) \leq 1$ (Ω) [13]. It's worth noting that the iterative activity produced by self-loop is not contained in the initial process model.

2) OPTIMIZATION OF REPAIR

The real-time repair proposed in this section can repair each unfitted behavioral pattern independently. The fitness and precision of a dataset depend on the total costs

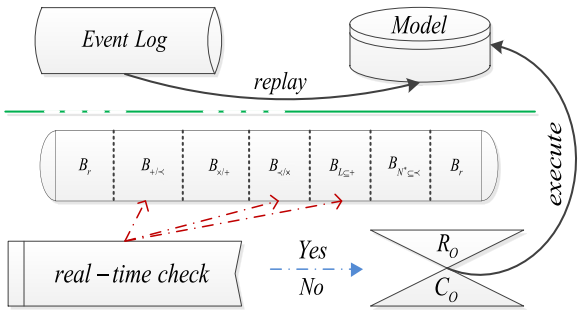

FIGURE 8. The principle of real-time repair.

of repairable and accurately repairable deviations in all behavioral patterns, respectively. The method of detection described in Section III checks for the iterative insert deviation in the given unfitted behavioral pattern. The repair and configuration operations can be switched to optimize the behavioral patterns according to the different results [27].

The repair operation is executed to repair the specified behavioral pattern, i.e., $R_O(IBP)$. $F(R_O(IBP)) = 1$ in any case, but $P(R_O(IBP)) < 1$ is affected by $\hbar$. On the contrary, the configuration operation is executed to repair the other behavioral pattern, i.e., $C_O(N - IBP)$. $P(C_O(N - IBP)) = 1$ can be obtained by λ and $F(C_O(N - IBP)) = 1$.

The improvements in real-time repair are as follows: i) all deviations can be replayed by $M - R(D)$ as compared with $C_O(D)$, i.e., Υ. Only if $\sum_D |IEsert\ell(e)|$ is zero does $F(C_O(D))$ equal $F(M - R(D))$. ii) $\sum_{N-IBP} |N - IEsert\ell(e)|$ can be accurately replayed by $M - R(D)$ as compared with $R_O(D)$, that is, Ξ. Only if $\sum_{N-IBP} |N - IEsert\ell(e)|$ is 0 does $P(R_O(D))$ equal $P(M - R(D))$. Therefore, the proposed method can improve precision as much as possible under the premise of perfect fitness. Table 2 presents the measure formulas and symbols of various repair methods.

Fig. 8 describes the basic principle of real-time repair. It checks the unfitted behavioral patterns using iterative insert deviations. The repair operation is executed if the result returns an affirmative (i.e., the pattern contains the iterative insert deviation). By contrast, the configuration operation is executed. As shown in Fig. 8, B_r represents the fitted behavioral pattern, $B_{+/\prec}$, $B_{\prec/\times}$, and $B_{L\subseteq+}$are the unfitted behavioral patterns with insert deviations, and $B_{\times/+}$ and $B_{N^*\subseteq\prec}$ are unfitted behavioral patterns with skip deviations. Thus, no check is executed in $B_{\times/+}$ and $B_{N^*\subseteq\prec}$.

B. REAL-TIME REPAIR OF DIFFERENCES PRODUCED BY BEHAVIOR PROFILE

The behavior profile in the business process can be included in three behavioral relationships: concurrency, causality, and conflict. This section introduces real-time repair to the unfitted behavioral pattern produced by the behavioral relations above.

TABLE 2. Formulas and symbols.

Equation No.	Formula
P	$\frac{(\sum_{n=1}^{N}\sum_{i=1}^{\lvert E_{C_n}\rvert}\frac{\Delta\lvert 1(e)\rvert}{\Delta\lvert\lambda(\sigma(1(e)))\rvert}\mathrm{g}\lvert C_n\rvert)}{\lvert L\rvert}$
φ	$\frac{\Delta\lvert N\text{-}IEsert1(e)\rvert}{\Delta\lvert\lambda(\sigma(1(e)))\rvert}=\frac{1}{\infty}=0$
Ω	$C\cos t=\sum_D\lvert IEsert1(e)\rvert$
F	$1-\frac{C\cos t}{\lvert L\rvert+\lvert M_L\rvert}=1-\frac{\sum_D\lvert Esert1(e)+\sum_D Skip\lambda(t)\rvert}{\lvert L\rvert+\lvert M_L\rvert}$
$P(R_O(IBP))$	$\frac{(\lvert L_{IBP}\rvert-\sum N\text{-}IEsert1(e)_{IBP})}{\lvert L_{IBP}\rvert}\le 1$
$\hbar$	$\Delta\lvert N\text{-}IEsert1(e)_{IBP}\rvert=1\wedge\Delta\lvert\lambda(\sigma(1(e)_{IBP}))\rvert=\infty$
$P(C_O(N\text{-}IBP))$	$\frac{\lvert L_{N\text{-}IBP}\rvert}{\lvert L_{N\text{-}IBP}\rvert}=1\ \ (\sum\lvert IEsert1(e)\rvert\notin N\text{-}IBP)$
D	$\Delta\lvert N\text{-}IEsert1(e)_{N\text{-}IBP}\rvert=1\wedge\Delta\lvert\lambda(\sigma(1(e)_{N\text{-}IBP}))\rvert=1$
$M\text{-}R(D)$	$R_O(\sum IBP)\cup C_O(N\text{-}IBP)$
$F(M\text{-}R(D))$	$F(R_O\sum IBP)+F(C_O(\sum N\text{-}IBP))=1\ \ (C\cos t=0)$
$F(C_O(D))$	$1-\frac{\sum_D\lvert IEsert1(e)\rvert}{(\lvert L\rvert+\lvert M_L\rvert)}$
$P(C_O(D))$	$\frac{(\lvert L\rvert-\sum_D\lvert IEsert1(e)\rvert)}{(\lvert L\rvert-\sum_D\lvert IEsert1(e)\rvert)}=\frac{\lvert L'\rvert}{\lvert L'\rvert}=1$ (L is preprocessed as L', i.e., the iterative activity produced by self-loop in the event log is removed)
$P(M\text{-}R(D))$	$\frac{(\lvert L\rvert-\sum_{IBP}N\text{-}IEsert1(e))}{\lvert L\rvert}$
$P(R_O(D))$	$\frac{\lvert L\rvert-(\sum_{IBP}\lvert N\text{-}IEsert1(e)\rvert)+\sum_{N\text{-}IBP}N\text{-}IEsert1(e))}{\lvert L\rvert}$
$\sum_D\lvert Esert1(e)\rvert$	$\sum_D\lvert IEsert1(e)\rvert+\sum_D N\text{-}IEsert1(e)$
$C\cos t$	$\sum_D\lvert Esert1(e)\rvert+\sum_D\lvert Skip\lambda(t)\rvert$
Υ	$F(M\text{-}R(D))=1\ge F(C_O(D))$
Ξ	$F(M\text{-}R(D))\ge P(R_O(D))$

Symbol	Annotation
F	the formula of fitness
P	the formula of precision
$\Delta\lvert 1(e)\rvert$	the maximal number of occurrences of an insert deviation in the event log
$\Delta\lvert\lambda(\sigma(1(e)))\rvert$	the maximal number of occurrences of an insert deviation in the process model
$conf_c$	conflict substructure of insert deviation and invisible transition ($(Esert1(e),\tau)_\times$)
$\sum_D\lvert Esert1(e)\rvert$	the total cost of insert deviations
$\sum_D\lvert Skip\lambda(t)\rvert$	the total cost of the skip deviations
$C\cos t$	the total cost of the deviations
D	dataset($D=\sum IBP\cup\sum N\text{-}IBP$)
BP	behavioral pattern
IBP	the unfitted behavioral pattern containing the iterative insert deviation
$N\text{-}IBP$	the unfitted behavioral pattern without the iterative insert deviation
$\lvert L_{IBP}\rvert/\lvert L_{N\text{-}IBP}\rvert$	the total number of events in IBP / $N\text{-}IBP$
$\sum_D\lvert IEsert1(e)\rvert$	the total cost of all iterative insert deviations in D
$\sum_D\lvert N\text{-}IEsert1(e)\rvert$	the total cost of all non-iterative insert deviations in D
$\sum_{IBP/N\text{-}IBP}\lvert N\text{-}IEsert1(e)\rvert$	the total cost of all non-iterative insert deviations in IBP / $N\text{-}IBP$

1) UNFITTED BEHAVIORAL PATTERN PRODUCED BY CONCURRENCY AND CAUSALITY

The concurrent/causal pattern refers to the concurrent behavioral relationship included in the pattern of the event log, whereas the causal behavioral relationship is included in the corresponding pattern of the process model [28]. Otherwise, no repair is needed. The following two operations can be interchangeably performed in the concurrent/causal pattern according to the results of checking for iterative insert

deviation: i) the occurrence and location of iterative insert deviation are uncertain; therefore, the repair operation needs to be implemented according to the deviation-related information detected by the optimal alignment; ii) the configuration operation involves adding or hiding the configurable behavior to or from the process model by constructing a conflict substructure comprising configurable behavior and an invisible transition and through which each repaired deviation can occur once at most.

Example 1: The fitted and unfitted behavioral patterns between the event log L_3 and the initial net system N^* in Fig. 9 are identified during replay as follows:

L_3	$\{(ABCDEF),(ABDCEF)\}$			
L_3'	$\{(ABCDDDEF),(ABDCEF)\}$			
B_{L_3/N^*}	A	B	$B_{\prec/+}(B_{+/\prec}(CD)E)$	F

$B_{\prec+}$ doesn't need to be repaired, and the behavioral patterns used for repair are denoted by:

B_{L_3/N^*}	A	B	$B_{+/\prec}(CD)$	E	F

According to the different results of checking for the iterative insert deviation in the unfitted behavioral pattern, two methods of repair are described in Fig. 9: i) the event log L_3 is changed into L_3', whereas the fitted and unfitted behavioral patterns are unchanged. Iterative insert deviations and non-iterative insert deviation are produced by the unfitted patterns between N^* and L_3' according to definition 5, that is, $IEsertD^2$ and $EsertD$. $IEsertD^2$ and $EsertD$ at different places, and regarded as two repair activities [29]. The iterative insert deviations are included in $B_{+/\prec}(CD)$, so two repair activities are inserted into N^* using two self-loops. The red dashed line and the solid green line represent the repair operations when the given pattern contains the iterative insert deviations. $EsertD$ cannot be accurately repaired using self-loop insert. ii) The configuration operations are executed when $L_3' = L_3$. Two constructed substructures are configured on the net system N^*, as shown by the dashed black line and the solid green line shown in Fig. 9 [30]. The iterative insert deviation is not included in $B_{+/\prec}(CD)$, allowing all insert deviations to be accurately repaired by the configuration operations. The green places represent locations in common between the configuration operation and the repair operation.

2) UNFITTED BEHAVIORAL PATTERN PRODUCED BY CONCURRENCY AND CONFLICT

The pattern of concurrency/conflict refers to the behavioral relationship in the pattern of the event log as concurrency, while the behavioral relationship in the corresponding pattern of the process model is identified as conflict. Otherwise, it is a pattern of conflict/concurrency. As this pattern contains only the skip deviation, the improvements in precision and fitness are identical for both operations. The pattern

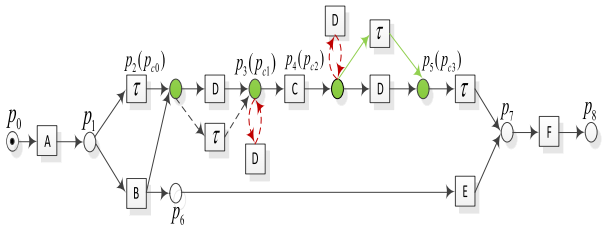

FIGURE 9. Repaired net-work system based on concurrent/causal behavior pattern.

of concurrency/conflict can switch between the operations depending on whether the unfitted behavioral pattern contains the iterative insert deviation. The deviation is diverse when the causality/conflict pattern contains an iterative insert deviation; thus, the repair operation should be performed based on the deviation-related information detected by the optimal alignment. Otherwise, the configuration operation is executed. Because the insert deviation is not included in the conflict/concurrency pattern, both operations can have the same effect on it in any case. This is denoted by $R_O \Leftrightarrow C_O(B_{(a,a')\times/+}) = Skip/Hide(\lambda(t), \lambda(t'))$ (i.e., the behaviors on the two branches are skipped separately).

Example 2: The fitted and unfitted behavioral patterns between the initial net system N^* in Fig. 10 and the event log L_4 are considered as follows:

L_4	$\{(ABCDEFG),(ACBDEFG)\}$		
L_4'	$\{(ABCCCDEFG),(ACBCCDEFG)\}$		
B_{L_4/N^*}	A	$B_{\times/+}(B_{\prec/\prec}(B_{+/\times}(BC)B_{L_4/N^*}(D))B_{\prec/\prec}(EF))$	G

According to the different results from checking for the iterative insert deviation, two operations can be used to repair $B_{+/\times}(BC)$: i) the event log L_4 is changed to L_4'. The iterative and non-iterative insert deviations are produced, i.e., $IEEsertC^3$ and $EsertC$. The iterative insert deviation is included in $B_{+/\times}(BC)$, after which the two repair activities are inserted into N^* using two self-loops. The dashed red line in Fig. 10 indicates repair operations of $EsertC$ and $IEEsertC^3$, respectively. $EsertC$ cannot be accurately repaired using a self-loop insert. ii) The configuration operation is executed when $L_4' = L_4$. The black dashed line shows the configuration of the conflict substructure in the configured replay graph. The iterative insert deviation is not included in $B_{+/\times}(BC)$. Thus, all insert deviations can be accurately repaired by the configuration operation. The solid green line in Fig. 10 shows the repair made to $B_{+/\times}(B_{\prec/\prec}(B_{+/+}(BC)B_{L/N^*}(D)B_{\prec/\prec}(EF))$, and $B_{+/\prec}(BC)$ has been repaired as $B_{+/+}(BC)$. The insert deviation is not present in this behavioral pattern, which allows all deviations to be accurately repaired by the repair or configuration operation. The black place represents the location of the configuration operation, and the blue and purple places connect configurable and common activities between repairable activities and configurable activities, respectively.

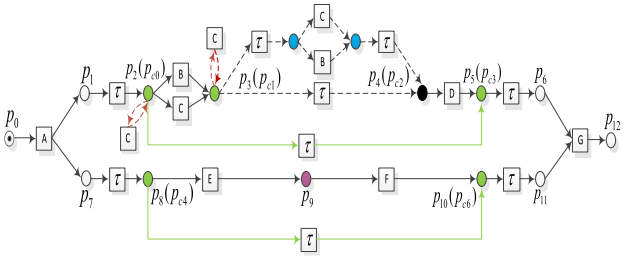

FIGURE 10. Repaired net-work system based on the concurrent/conflicting behavioral pattern.

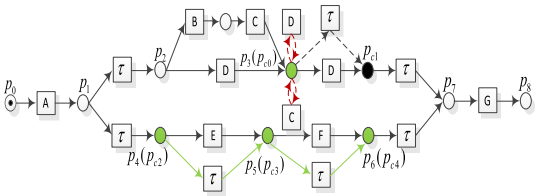

FIGURE 11. Repaired net-work system based on the causal/conflicting behavior pattern.

3) UNFITTED BEHAVIORAL PATTERN PRODUCED BY CAUSALITY AND CONFLICT

The pattern of causality/conflict refers to a behavioral relationship in the pattern of the event log characterized by causality, while the behavioral relationship in the corresponding pattern of the process model is conflict. Otherwise, it is a pattern of conflict/causality. The deviations or configurable behaviors produced by the causality/conflict pattern originate from the branch with the least number of activities in the conflict pattern. The repair operation is performed on the process model based on the deviation-related information detected by the optimal alignment when the pattern of concurrency/conflict includes the iterative insert deviation. Otherwise, the configuration operation is executed according to configurable behavior in the configured replay graph. Because the conflict/causality pattern does not contain insert deviation, the effect of both operations is the same (i.e., behaviors on the two branches of the conflict pattern are skipped separately).

Example 3: The fitted and unfitted behavioral patterns between the initial network system N^* in Fig. 11 and the event log L_5 are recorded as follows:

L_5	$\{(ABCDEG),(ABCDFG)\}$		
L_5'	$\{(ABCCCDDDEG),(ABCDFG)\}$		
B_{L_5/N^*}	A	$B_{\prec/\times}(B_{\prec/\times}(B_{\prec/\prec}(BC)D)B_{\times/\prec}(EF))$	G

According to the results of checking for the iterative insert deviation, $B_{\prec/\times}(BC)$ can be repaired in the following two ways: i) The event log L_5 is changed to L_5'. The iterative and non-iterative insert deviations are produced, i.e., $\{IEsertC^2, IEEsertD^3\}$ and $EsertD$. The iterative insert deviations are included in $B_{\prec/\times}(B_{\prec/\prec}(BC)D)$, after which two repair activities are inserted into N^* using two self-loops. . The dashed red line in Fig. 11 indicates repair operations. $IEEsertD^3$ and $EsertD$ at the same places, which are regarded as a repair activity. $IEsertC^2$ and $IEEsertD^2$ can be accurately repaired, whereas $EsertD$ cannot be accurately repaired. ii) The configuration operation is executed when $L_5' = L_5$. The black dashed line represents $Add(D)$ as a substructure to be configured in N^*. The iterative insert deviation is not included in $B_{\prec/\times}(B_{\prec/\prec}(BC)D)$, which allows the insert deviation to be accurately repaired by the configuration operation. In Fig. 11, the solid green line represents the repair of the unfitted behavioral pattern that contains only the skip deviation denoted by $B_{\times/\prec}(EF)$. The deviations or configurable activities in this unfitted behavioral pattern are $\{SkipE \cup SkipF\}/\{Hide(E) \cup Hide(F)\}$, and cannot be skipped as a whole because of selective occurrence.

Algorithm 1 describes how to construct the operations of configuration and repair in the five unfitted patterns: $B_{+/\prec}$, $B_{+/\times}/B_{\times/+}$, and $B_{\prec/\times}/B_{\times/\prec}$. The sets of configuration operations and repair operations are assumed to be empty (line 1). The configuration operations (lines 6–8) and repair operations (lines 8–10) are performed on the patterns of $B_{+/\prec}$, $B_{+/\times}/B_{\times/+}$ (lines 13–15 and 15–17, respectively), and $B_{\prec/\times}/B_{\times/\prec}$ (lines 20–22 and 22–24, respectively). The sets of configuration and repair operations of the unfitted behavioral patterns formed due to causality, concurrency, and conflict are returned. The judgment formula is then used to choose the repair or the configuration operation in the given behavioral pattern, denoted by $I\cos t_{BP} = 0$ (ψ is the check function that obtains the total cost of iterative insert deviations in the given pattern). The occurrences of $IEEsert\ell(e)$ and $IEsert\ell(e)$ are random; therefore, all self-loop insert formations are related to repair operations of patterns and contain insert deviations, where this is denoted by $R_O(B_{(+/\prec)(+/\times)(\prec/\times)}) \leftarrow \chi$. The repair operations of patterns without an insert deviation are interchangeable with configuration operations.

4) REAL-TIME REPAIR TO DIFFERENCES PRODUCED BY BEHAVIOR INCLUSION

The behavior inclusion pattern refers to behavior included in a pattern that cannot be captured on the process model but can be exactly recorded in the event log. On the contrary, only behavior that disables the event log from being replayed is considered. All activities in the complete behavior inclusion pattern are constructed into a substructure according to behavior type for repair. The partial behavior inclusion in a conflict pattern means that the event log contains a conflict pattern and the process model can capture only behavior on one branch of this pattern. Otherwise, it does not need to be repaired. The resulting differences are a combination between common and specific behaviors in the event log. There can be uncertain occurrences of an iterative insert deviation in an unfitted behavioral pattern. The relevant operation is thus performed according to check result of iterative insert deviation.

Algorithm 1 Real-Time Repair to Differences Produced by Behavior Profile

Input: net N,Log L,R($\wp_L$),R($\wp_N$), ψ, A_{op}
Output: C_O, R_O

1: $C_O \leftarrow \emptyset, R_O \leftarrow \emptyset$
2: **for each** $e_i \in L$ **do**
3: **for each** $t_j \in T$ **do**
4: **for each** $a_k \in \aleph \cup \mathbb{Z}$ **do**
5: **if** $\ell(e_i) = \lambda(t_j) = a_k \wedge B_{+\perp}(a_1, a_2)$ **then**
6: **if** $\psi(I\cos t) = 0$ **then**
7: $C_O \leftarrow (Add(a_2) \cup Hide(a_2))$
8: **else**
9: $R_O \leftarrow skip(a_2)) \cup \chi(sl - Esert(a_2), sl - IEsert(\ell(e_i)))$
10: **end**
11: **end**
12: **if** $\ell(e_i) = \lambda(t_j) = a_k \cup a'_k \wedge B_{+/\times}/B_{\times/+}(a_1, a_2)$ **then**
13: **if** $\psi(I\cos t) = 0$ **then**
14: $C_o \leftarrow N_{A/H(a_1,a_2)}$,
15: **else**
16: $R_O \leftarrow \chi(sl - Esert(a_1/a_2), sl - IEsert(\ell(e_i)))/R_O \Leftrightarrow C_O$
17: **end**
18: **end**
19: **if** $\ell(e_i) = \lambda(t_j) = a_k \wedge B_{\perp\backslash\times}/B_{\times/\perp}(a_1, a_2)$ **then**
20: **if** $\psi(I\cos t) = 0$ **then**
21: $C_O \leftarrow (Add(a_1/a_2)/(Hide(a_1) \cup Hide(a_2)))$
22: **else**
23: $R_O \leftarrow \chi(sl - Esert(a_1/a_2), sl - IEsert(\ell(e_i)))/R_O \Leftrightarrow C_O$
24: **end**
25: **end**
26: **end**
27: **end**
28: **end**
29: **return** C_O, R_O

Example 4: According to the different types of behavior inclusion patterns, the fitted and unfitted behavioral patterns between L_6 and the initial net system N^* in Fig. 12 can be set as follows:

L_6	{(*ABCFHIJKL*),(*ABDEGIHJKL*)}			
L_6'	{(*ABCCCFHIL*),(*ABDEGGGIHHHL*)}			
B_{L_6/N^*}	*A*	*B*	$B_{L_6\subseteq^*_\times}((C)B_{L_6\subseteq_\prec}(ED)), B_{L_6\subseteq^*_\times}(GF), B_{L_6\subseteq^*_\times}(HI), B_{N^*\subseteq_\prec}(JK)$	*L*

Because $B_{L_6\subseteq^\bullet_\times}((C)B_{L_6\subseteq_\prec}(ED))$ is the unfitted behavioral pattern caused by partially conflicting behavior inclusion, it contains both insert and skip deviations. According to the complete information on the deviation and the results of checking for the iterative insert deviation, two methods are used to repair the unfitted behavioral patterns in Fig. 12: i) L_6 is changed to L'_6. The iterative and non-iterative insert deviations are produced, i.e., ($IEsertC^2$, $IEEsertG^3$, $IEEsertH^3$) and ($EsertD$, $EsertE$, $EsertF$, $EsertI$). Seven repair activities are inserted into N^* using seven self-loops. The red dotted line and the solid green line in Fig. 12 describe the repair operations. In this case, all non-iterative insert deviations cannot be accurately repaired. ii) The configuration operations are executed when $L_6 = L'_6$. The black dashed line and the solid green line describe the configuration operations on N^*. The iterative insert deviation is not included in the unfitted behavior pattern, so all insert deviations can be accurately repaired by the configuration operations. The solid green line in Fig. 12 shows the repair made to $B_{N^*\subseteq_\prec}(JK)$. No insert deviation is contained in $B_{N^*\subseteq_\prec}(JK)$. Thus, all deviations can be accurately repaired by the repair or the configuration.

Algorithm 2 Real-Time Repair to Differences Produced by Behavior Inclusion

Input: net N,Log L,R($\wp_L$),R($\wp_N$),icost
Output: C_O, R_O

1: $C_O \leftarrow \emptyset, R_O \leftarrow \emptyset$
2: **for each** $a_k \in \aleph \cup \mathbb{Z}$ **do**
3: **if** $B_{L/N^*+/\perp/\times}((a_1, a_2) \in R(\wp_L)/R(\wp_{N^*})$ $\wedge B_{N^*/L+/\perp/\times}(a_1, a_2) \in R(\wp_{N^*})/R(\wp_L))$ **then**
4: **if** $\psi(i\cos t) = 0$ **then**
5: $C_O \leftarrow (\tau \perp ((N_{A\perp}/N_{A+}/N_{A\times}), \tau)_\times \perp \tau)/(\tau \perp (\tau, (N_{H\perp}/N_{H+}/N_{H\times}))_\times \perp \tau)$
6: **else**
7: $R_O \leftarrow \chi(sl - IEsert(\ell(e_i)), N_{sl-E(a_1,a_2)+}, N_{slE(a_1,a_2)\perp}, N_{sl-E(a_1,a_2)\times})/(N_{S\perp}/N_{S\times}/N_{S+})$
8: **end**
9: **end**
10: **if** $B_{L\times}(a_1, a_2) \in R(\wp_L) \wedge (a_1 \in R(\wp_{N^*}))$ **then**
11: **if** $\psi(I\cos t) = 0$ **then**
12: $C_O \leftarrow (Add(a_2) \cup Hide(a_1))$
13: **else**
14: $R_0 \leftarrow \chi(sl - Esert(a_2), sl - IEsert(\ell(e_i))) \cup skip(a_1)$
15: **end**
16: **end**
17: **end**
18: **return** C_O, R_O

Algorithm 2 describes the configuration and repair operations of various behavior inclusion patterns. First, the sets of configuration and repair operations are set to empty (line 1); second, the configuration operations (lines 4–6) and repair operations (lines 6–8) of the three types of complete behavior inclusion patterns are constructed. Third, the configuration (lines 11–13) and repair operations (lines 13–15) of the partial behavior inclusion in the conflict pattern are constructed. Finally, the sets of configuration and repair operations produced by the behavior inclusion patterns are returned.

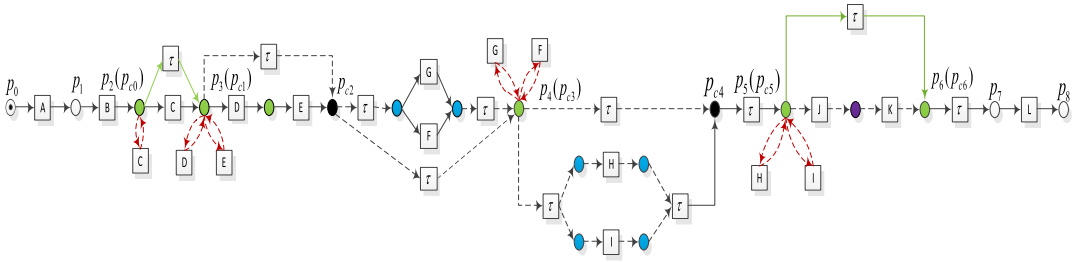

FIGURE 12. Repaired network system based on the behavior inclusion pattern.

C. SUBSTRUCTURE OF DEVIATIONS

Precision needs to be improved in the behavioral pattern containing an iterative insert deviation. The directly following non-iterative insert deviations detected by the optimal alignment can be constructed into a substructure, i.e. the several deviations are regarded as a substructure. Thus, the degree of accurate replay is improved in this case (i.e., the number of events in the event log that need to be accurately repaired are reduced, whereas the number of events that can be accurately repaired remains the same). $Esert(\ell(e_1), \ell(e_2), \ldots, \ell(e_n))_{+/\prec}$ represents the directly following non-iterative insert deviations that satisfy the concurrency/causality relationship. The size of the event log cannot be reduced by the conflict substructure; thus, the directly following non-iterative insert deviations are composed based only on concurrency and causality relationships.

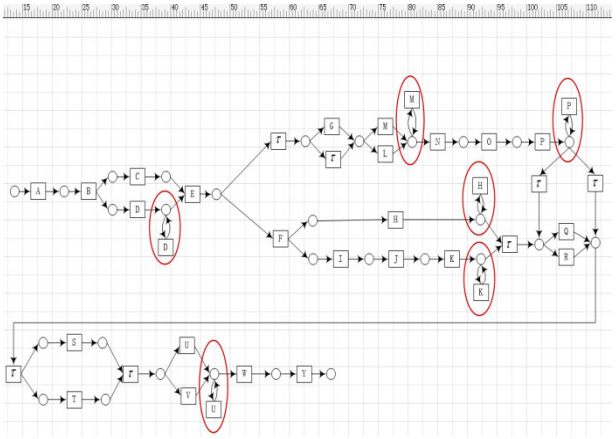

FIGURE 13. The real-life business process with the iterative insert activities.

V. EVALUATION

A. EXPERIMENTAL SETUP

This section describes the results of a comparison between the method proposed in this article and the existing method[14]. An experiment was performed on a 64-bit Win10 computer with Inter(R) Core(TM) i5-2.11 GHz, with 8 GB of memory space and JDK1.7. The method proposed here involved the following two improvements: i) the two operations were switched according to whether there was an iterative insert deviation in the unfitted behavioral pattern; ii) in case of an iterative insert deviation in the pattern, the directly following non-iterative insert deviations were constructed into a substructure based on different behavioral relationships.

1) TOOL SUPPORT

The performance of the existing method was verified using the Prom framework. To ensure fairness of the experimental results, we wrote a plug-in called M-repair into JAVA to verify the fitness and precision of the repaired dataset according to the different functions implemented by the two methods. This plug-in is publicly available on Google Cloud, and a set of data have been attached for testing.[2] The experiment used JAVA to write a plug-in PSLG similar to CPN-Tools, which automatically generated event logs in txt format or xes format by taking advantage of the random triggering of activities in the business process.

2) DATASETS

The experiment used datasets from an artificial business process and a real-life business process for evaluation, denoted respectively by D_a and D_r. The real-life business process was from a life insurance company in China. It described processes from the application of the policy to the signing of the contract. The set of firing sequences consistent with these behaviors was obtained by simulating the business process. The unfitted behavioral patterns in the datasets were divided into the following three types: i) unfitted behavioral patterns caused by concurrency/causality/conflict/single activity, ii) behavior inclusion patterns, and iii) unfitted behavioral patterns caused by the cycle. To guide the normal conduct of the experiment, we changed the firing sequences and obtained the given process model according to the three unfitted behavioral patterns above. The three corresponding event logs were automatically extracted from each business process based on random self-loop insert activities using the PSLG plug-in. Each business process was related to the three process models. An event log and a process model formed a dataset. The experiment consisted of three business processes, for a total of nine event logs and given processes. Therefore, the set of datasets of the experiment contained artificial datasets,

[2]https://drive.google.com/file/d/1iFsfW4XJJRrXPk8s7rw8xO8K40OW0-LQ/view?usp=sharing.

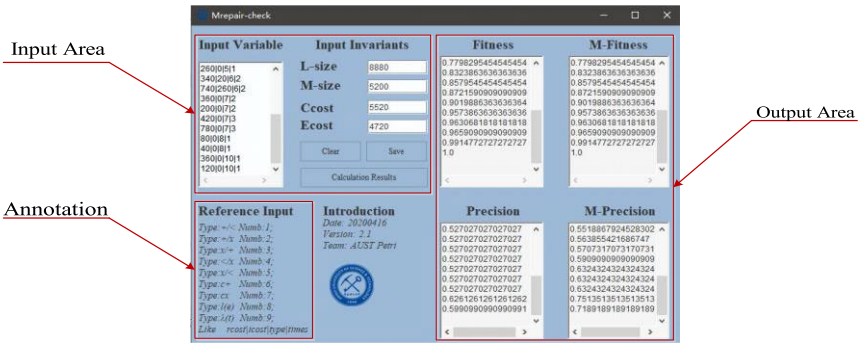

FIGURE 14. The operation interface of M-repair-check plugin.

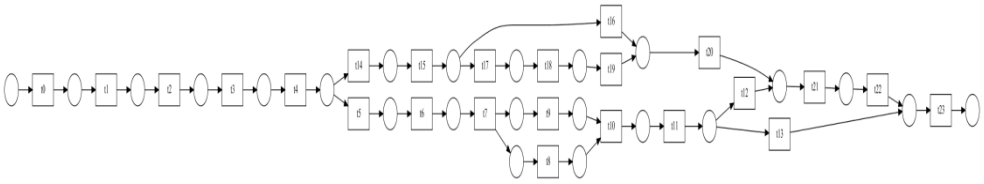

FIGURE 15. The given process model before real-time repair.

TABLE 3. Information of datasets.

Dataset	D_{a1}	D_{a2}	D_{a3}	D_{r1}	D_{r2}	D_{r3}	$D_{\perp 1}$	$D_{\perp 2}$	$D_{\perp 3}$
Activity	21	21	22	25	25	25	23	25	25
$\|L\|(10^2)$	520	366	482	464	340	436	596	500	558
$\|M\|(10^2)$	240	220	482	338	464	246	224	320	344

real-life datasets, and datasets with loops. The automatically generated event log contained 500 cases, for a total of 10 representative cases selected from the initial event log to ensure that all events and the behavior relations between them as recorded in the event log were observed. The 10 representative execution sequences were repeated 200 times based on the number of occurrences of each sequence, so that the event log for each dataset contained 2000–2800 traces. In this way, a business process yielded three sets of sequences of execution from different event logs. These sequences, along with the set of initial firing sequences of the business process, were optimally aligned to form three datasets. The information of nine datasets in experiment were recorded in table 3.

The real-life business process of the experiment was shown in Fig. 13. The red ellipses indicate seven observable activities randomly inserted into the business process using seven self-loops.

3) EXPERIMENTAL STEPS AND CRITERIA

The experimental steps consisted of the following three parts: i) the replay fragments in the dataset were divided according to the different behavioral relationships under the given constraints. Then, the unfitted behavioral patterns produced by them were accurately identified. ii) The plug-in M-repair determined whether the iterative insert deviation was present in the unfitted behavioral pattern and chose the appropriate operation for the repair process model accordingly. iii) The fitness and precision of the repaired datasets were automatically measured. Because real-time repair required switching operations, $I \cos t_{BP} > 0$ was used to identify the iterative insert deviation of the behavioral pattern in the plug-in.

4) PLUG-IN OPERATION INTERFACE

Fig. 14 shows the display interface of a dataset executed in the plugin M-repair. The input parameters include the input variables and invariables. The button (called ''calculation results'') was used to obtain the fitness and precision of the dataset repaired by the two methods. In Fig. 14, the upper-left corner shows the area of various input parameters required by the experiment, the left side of the lower-left corner contains detailed comments on the types of variables, and the right side shows the area of outputs of the values of fitness and precision obtained when the dataset was repaired by the two methods.

5) EXAMPLES OF THE PRE- AND POST-REPAIR PROCESS MODELSES

A real-life business process is used as an example and an observable activity is generated by the self-loop. The event logs were extracted from the business process using the PLSG plug-in. Unfitted behaviors were obtained between the given process model and the event log. The initial process model and that after real-time repair are shown in Figs. 15. It was mined by the Prom framework. The fitness and precision were 1 and 0.8 when the event logs were replayed, as shown in Fig. 15. Fig. 16 extracts the sub-model containing the repair operation for a behavioral pattern and the configuration operation for the other behavioral pattern (t_{15} is the invisible transition). This sub-model is represented by the purple box.

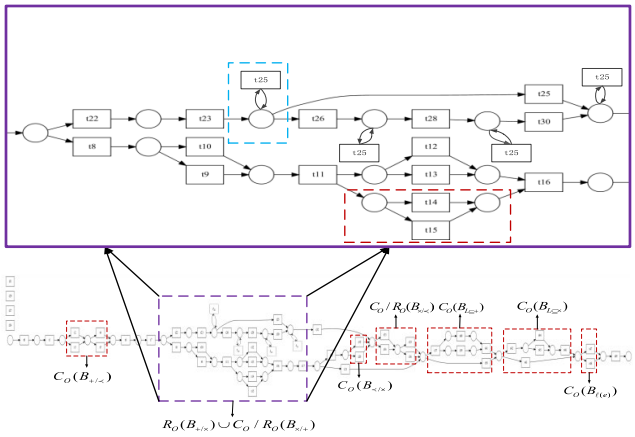

FIGURE 16. The process model after real-time repair based on real-life event log.

The conflict substructure constructed by t_{14} and t_{15} is the execution of the configuration operation, whereas the self-loop insert transition t_{25} represents the repair operation of the observable deviation. The repair and configuration operations are marked with blue and red dotted boxes in Fig. 16, respectively. The two operations can be alternatively performed according to whether there is iterative insert deviation in the unfitted behavioral pattern. The dataset contained only one unfitted behavioral pattern with an iterative insert deviation; there were thus seven configuration operations and a repair operation in the repaired process model. The behavioral patterns without an iterative insert deviation repaired by the configuration operations could entirely be accurately replayed. On the contrary, the non-iterative insert deviations could not be accurately replayed using the existing or the proposed method. Therefore, the loss of precision in the proposed method was dependent only on deviations that occurred once in the behavioral pattern containing an iterative insert deviation.

B. ANALYSIS OF RESULTS: FITNESS IS 1 AFTER REPAIR

To ensure that the fitness between the repaired model and the event log was always one, all observable deviations were inserted into the model using self-loops. M-Repair used different metrics to calculate the fitness and precision of the real-time repair and repair [17]. The formula for the behavioral pattern is shown in Table 4. $R\cos t_{BP}$, $E\cos t_{BP}$, $I\cos t_{BP}$, and $N-I\cos t_{BP}$ represent the total cost of the deviations, cost of the insert deviations, cost of the iterative insert deviations, and the cost of the non-iterative insert deviations in the unfitted behavioral pattern, respectively.

1) EVALUATION OF BUSINESS PROCESS WITHOUT LOOP

Six datasets without loops were derived from the artificial and real-life business processes. The numbers of iterative insert deviations in the datasets were different, because of which there was a difference in improvements in precision yielded by the proposed method and the self-loop insert/skip. The initial value of precision was the value that all insertion deviations cannot accurately repair and that can potentially be improved with the repair of each behavior pattern.

- **Results on the artificial datasets**

Fig. 17 describes the results of implementation of the three datasets on the artificial business process. The squares and triangles represent the values of precision repaired by the existing method and the proposed method, which are denoted by P and M-p, respectively. The improvements effected by M-p in Figs. 17 (a) and 17 (c) were significantly lower than those in Fig. 17 (b), while those in Fig. 17 (a) were superior to those Fig. 17 (c). The former result obtained because the unfitted behavioral patterns with the iterative insert deviation only accounted for 20% of the total number of patterns in Fig. 17 (b). Thus, the precision of all other behavioral patterns were improved owing to the configuration operations in M-P. Fig. 17 (b) shows precision 21% higher than those of the existing method. The latter result obtained because n was three in the causal behavior inclusion pattern of the dataset as described in Fig. 17 (a). We constructed the three datasets directly following non-iterative insert deviations into the substructure. Precision improved by 6% (Fig. 17 (a)) relative to that of the existing method and was 2% higher than that in Fig. 17 (c). The costs of repair of the two methods were the same. $i\cos t^+$ and $r\cos t^+$ represent the increasing costs of the iterative insert deviations and the repairable deviations, respectively.

- **Results on real-life datasets**

Fig. 18 depicts the experimental results of the three datasets applied to the business process considered. They differed in terms of improvements in precision after being repaired when the two methods were used. The improvements of P effected by M-p, shown in Figs. 18 (a) and (c), were significantly lower than that shown in Fig. 18(b). The improvement of P by M-p in Fig. 18 (a) was 3% lower than Fig. 18 (c). The reason for the former result is similar to those shown in Figs. 17 (a), (b), and (c). In Fig. 18 (b), the unfitted behavioral patterns with iterative insert deviations account for only 20% of the dataset. Precision was improved by 22% compared with the existing method. The ratio of the cost of iterative insert deviations in terms of total cost is denoted by Ic^+/Rc^+. The reason for the latter was Ic^+/Rc^+ in Fig. 18 (a) slightly higher than Fig. 18 (c). Ic^+/Rc^+ was inversely proportional to the improvement in precision in general. The costs of repair of the two methods were the same.

- **Results on datasets with loops**

Fig. 19 describes the experimental results of the three datasets in case of loops. Differences were observed in terms of the improvement in precision. Compared with P, the value of M-P (Fig. 19 (c)) recorded the highest improvement. This is because unfitted behavioral patterns with iterative insert deviations accounted for only 10% of this dataset. The precision was 33% higher than that of the existing method. The unfitted behavioral pattern produced by the cycle had the following three different repair-related effects: i) the path of

TABLE 4. The Formulas to measure repair performances.

Plug-in	Fitness	Precision	
Check	$I\cos t_{BP} \geq 0$	$I\cos t_{BP} \geq 0$	$I\cos t_{BP} = 0$
Repair	$1-\frac{C\cos t - R\cos t_{BP}}{\lvert L\rvert + \lvert M_L\rvert}$	$\frac{L - E\cos t + I\cos t_{BP}}{L-(R\cos t_{BP} - I\cos t_{BP})*((n-1)/n)}(B_{L\subseteq^+})$	$\frac{L - E\cos t + I\cos t_{BP}}{L-(R\cos t_{BP} - I\cos t_{BP})*((n-1)/n)}(B_{L\subseteq^+})$
Real-time repair	$1-\frac{C\cos t - R\cos t_{BP}}{\lvert L\rvert + \lvert M_L\rvert}$	$\frac{L - E\cos t + I\cos t_{BP}}{L-(R\cos t_{BP} - I\cos t_{BP})*((n-1)/n)}$	$\frac{\lvert L\rvert - E\cos t_{BP} + N\text{-}I\cos t_{BP}}{\lvert L\rvert}$

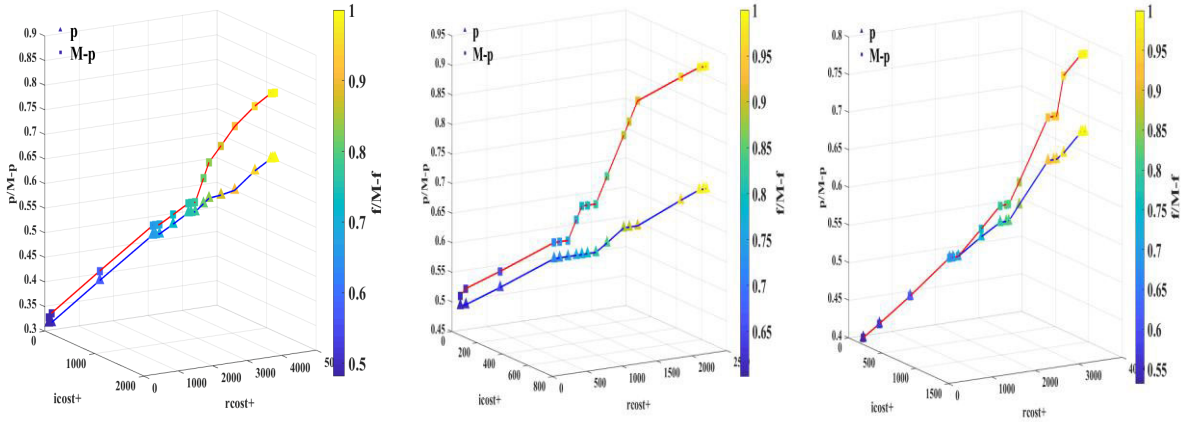

FIGURE 17. Results using the artificial datasets.

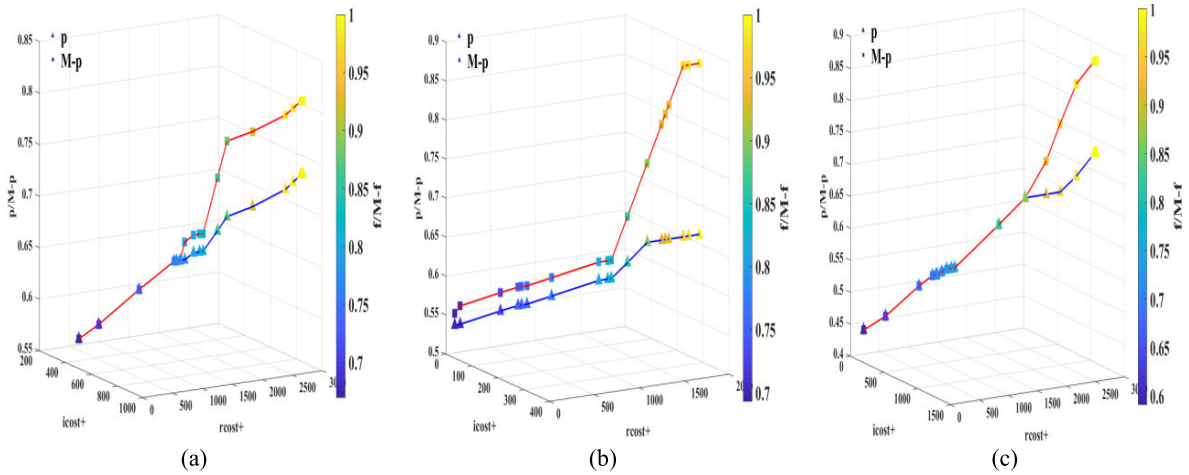

FIGURE 18. Results using the real-life datasets.

the loop existed only in the event log. In this case, if there were deviations only generated by cycle, the improvement in precision of both repair methods was the same (i.e., they both used invisible transitions to connect the loop from start to end). ii) The acyclic part was unfitted and could be repaired directly according to the operation of the acyclic part. iii) Behavior on the return path could not be replayed. At this time, the behavioral patterns on the return path could be divided, and the unfitted behavioral patterns could be repaired using the repair method used on the acyclic part. Accordingly, the unfitted behavioral patterns with iterative insert deviations accounted for 20% (Figs. 19 (a) and (b)). The iterative insert deviations divided by the total cost were 7% higher than in Fig. 19 (a); however, the difference in precision led to only a 1% difference in terms of improvement, because Fig. 19 (a) used only invisible transitions to repair the cycle, and there was no deviation in the acyclic part and return path. Thus, the deviations produced by this loop could be accurately repaired by two methods. M-p improved by 21% compared with P, while the result was 1% higher than that in Fig. 19 (b).

TABLE 5. Experiment results on different datasets. Experiment results on artificial datasets.

Experiment results on artificial datasets							
Dataset	fit	M-fit	pre	M-pre	Rcost$^+$	Icost$^+$	Ic$^+$/Rc$^+$
D_{a1}	1	1	0.70	0.76	40600	18200	45%
D_{a2}	1	1	0.71	0.92	24600	6600	27%
D_{a3}	1	1	0.69	0.73	34200	12200	36%
Experiment results on real-life datasets							
Dataset	fit	M-fit	pre	M-pre	Rcost$^+$	Icost$^+$	Ic$^+$/Rc$^+$
D_{r1}	1	1	0.74	0.81	26800	10000	37%
D_{r2}	1	1	0.67	0.89	22600	3200	14%
D_{r3}	1	1	0.74	0.84	28400	11200	39%
Experiment results on datasets with loop							
Dataset	fit	M-fit	pre	M-pre	Rcost$^+$	Icost$^+$	Ic$^+$/Rc$^+$
$D_{\downarrow 1}$	1	1	0.75	0.96	29000	6400	22%
$D_{\downarrow 2}$	1	1	0.71	0.91	35400	10200	29%
$D_{\downarrow 3}$	1	1	0.60	0.93	39400	7400	19%

The costs of repairs incurred by the two methods were the same.

The experimental results on the nine datasets are shown in Table 5. Ic^+/Rc^+ in D_{a1} was 9% higher than D_{a3}, whereas the improvement in precision in D_{a3} was 2% lower than that in D_{a1}, because there was a causal behavior-inclusion pattern in D_{a1}. The three non-iterative insertion deviations were constructed as a causal substructure. The value of Ic^+/Rc^+ of D_{r2} was 5% lower than that of $D_{\downarrow 3}$, whereas improvement in its precision was 10% lower. This is because non-iterative insert deviations in the acyclic part were repeatedly produced by the cycle. The precision of the first six datasets without loops improved by an average of $(6\% + 21\% + 4\% + 7\% + 22\% + 10\%)/6 \approx 12\%$, whereas that of the last three datasets with loops was $(21\% + 20\% + 23\%)/3 \approx 21\%$, according to Table 5.

C. ANALYSIS OF RESULTS: FITNESS IS UNCERTAIN AFTER DIFFERENT REPAIRS

The datasets were repaired using the proposed method and two existing methods. The two existing methods were as follows: i) the insert deviation was repaired by the self-loop described in Section V.B to ensure fitness in any case. The iterative insert deviation could be accurately repaired only by this method, so that precision was affected by the non-iterative insert deviation. This is denoted by R_O. ii) The configurable activity and invisible transition were combined into a conflict structure that was to the appropriate location in the process model. The iterative insert deviation could not be repaired, because of which the fitness of the dataset with iterative insert deviations could not be guaranteed by this method. This is denoted by C_O. Fitness and precision formulae of C_O were added to M-repair to implement by $F(C_O)$ and $P(C_O)$. The proposed method is denoted by $M-R$. The datasets in Fig. 18(a), Fig. 19(a), and Fig. 19(b) are used as examples to analyze the results below, and are denoted by D_{r1}, $D_{\downarrow 1}$, and $D_{\downarrow 2}$, respectively. It's worth noting that the iterative activity produced by self-loop was not contained in the initial process models of nine datasets.

1) PERFORMANCE ANALYSIS OF THREE REPAIRS WITH UNCERTAIN DEGREES OF FITNESS

Fig. 20 shows the results of repairs of D_{r1}, $D_{\downarrow 1}$, and $D_{\downarrow 2}$ using the above three methods. The main aim of model repair is to be able to completely replay the event log to the process model. Thus, fitness of 1 must first be guaranteed after repair. The results in D_{r1}, $D_{\downarrow 1}$, and $D_{\downarrow 2}$ (Figs. 20 (a), (b) and (c)) show that the precision of $M - R$ was higher than that of R_O by $P \uparrow \{7\%, 21\%, 20\%\}$. The fitness of $M - R$ and R_O were both one in D_{r1}, $D_{\downarrow 1}$, and $D_{\downarrow 2}$. The iterative deviations in the event log were removed, so the precision of C_O was one [13]. However, the fitness of C_O was 86%, 92% and 89% in D_{r1}, $D_{\downarrow 1}$, and $D_{\downarrow 2}$, respectively. Although the precision of C_O was higher than that of $M - R$, it needed to sacrifice the replay of iterative insert deviations. Therefore, $M - R$ was the optimal solution to instances of unfitted behavior in terms of fitness and precision in D_{r1}, $D_{\downarrow 1}$, and $D_{\downarrow 2}$.

2) RECALL, ACCURACY, AND NUMBER OF REPAIR ACTIVITIES

The recall and accuracy of the three methods were compared using the same number of repair activities. The number of repair activities was divided into nine groups and increased continuously according to the unfitted behavioral pattern in each dataset.

The recall represents the ratio of the cost of replay deviations to the total cost of deviations, and is denoted by $C\cos t_r/C\cos t$. The recall values of R_O and $M - R$ reached one while C_O could not in D_{r1}, $D_{\downarrow 1}$, and $D_{\downarrow 2}$ (Figs. 21 (a), (b) and (c)). The recall of C_O in D_{r1} was 22%, 16% less than those shown in $D_{\downarrow 1}$, and $D_{\downarrow 2}$, respectively. This is because all iterative insert deviations could not be replayed by C_O. Ic^+/Rc^+ in D_{r1} was much higher than those in $D_{\downarrow 1}$, and $D_{\downarrow 2}$.

Accuracy refers to the ratio of the cost of insert deviations that can be accurately repaired to the total cost of insert deviations and is denoted by $E\cos t_r/E\cos t$. The accurate repair of a deviation means that the similarity between the behavior following repaired and the original behavior is one. $M - R$ recorded the accuracies as shown in D_{r1}, $D_{\downarrow 1}$ and $D_{\downarrow 2}$ (Figs. 20 (a), (b) and (c)), of 62%, 90%, and 89%, respectively. The accuracies of R_O were 19%, 53%, and 53% less than those of $M - R$ shown inD_{r1}, $D_{\downarrow 1}$, and $D_{\downarrow 2}$, respectively. This is because Ic^+/Rc^+ of $D_{\downarrow 1}$ and $D_{\downarrow 2}$ was significantly lower than D_{r1}. The insert deviations accurately repaired by $M - R$ were divided into the following two types: i) iterative insert deviations; ii) all insert deviations of the unfitted behavioral pattern without iterative insert deviations (i.e., a part of non-iterative insert deviations). However, only the iterative insert deviations could be accurately repaired by R_O. The accuracy of C_O was one in Fig. 21 (a), (b), and (c). All insert deviations in the preprocessed event log were accurately repaired by C_O [13]. However, recall of C_O could not be guaranteed. Therefore, $M - R$ was the optimal solution to instances of unfitted behavior in terms of recall and accuracy, given

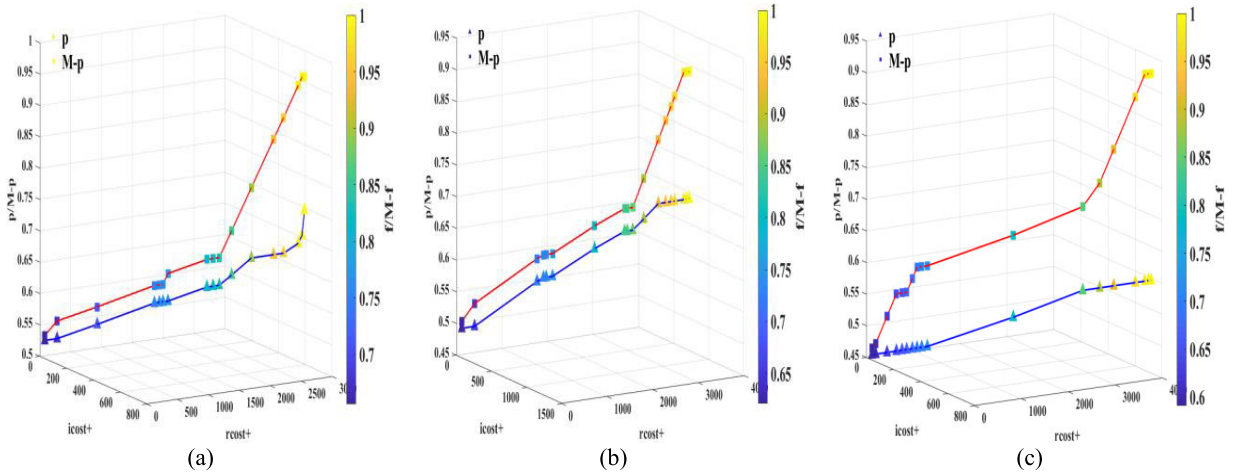

FIGURE 19. Results using the artificial and real-life data sets with loops.

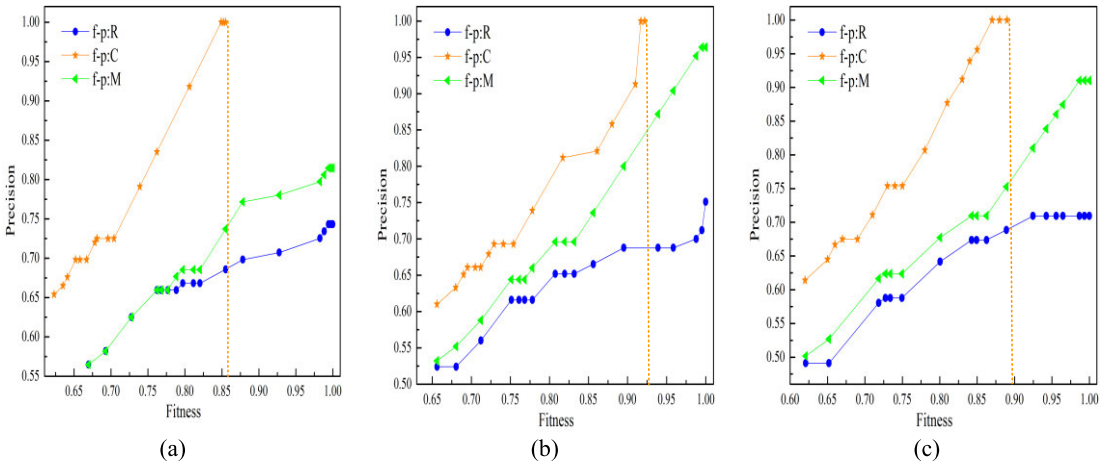

FIGURE 20. Results using the real-life datasets under uncertain conditions.

the same number of repair activities in D_{r1}, $D_{\downarrow 1}$, and $D_{\downarrow 2}$ (Figs. 21 (a), (b), and (c)).

The experimental results on the three datasets show that the real-time repair proposed in this paper is feasible and effective. It can improve precision as much as possible while guaranteeing fitness. The improvements in precision varied on different datasets for the following reasons: i) the number of unfitted behavioral patterns without iterative insert deviation in different datasets is different, ii) the numbers of substructures combined by the directly following insert deviations in different datasets were different, and iii) the types of cycles that needed to be repaired were different.

VI. RELATED WORK

The method proposed in this paper is closely related to the following three technologies: i) conformance check of business processes, ii) configuration optimization, and iii) model repair [31].

A. CONFORMANCE CHECK

According to the different types of deviations, the prevalent conformance check technology can be divided into two categories: i) the occurrence and location of the deviation were detected based on the optimal alignment between the event log and the process model; ii) the mismatched behavioral pattern was found based on the behavioral relationships between the event log and the process model [26]. The first check method used a heuristic algorithm, and trace replays were used to obtain an effective optimal alignment [32]. The costs of synchronous and asynchronous moves were set to zero and one, respectively (the cost of asynchronous movement with invisible transition was zero). The second method of conformance check involved generating a partial synchronization product between the event log and the process model by constructing the event structure. Then, the execution configuration set was expanded to discover the elements of deviation and their potential behavioral relations. By introducing types of deviations to the nine mismatched

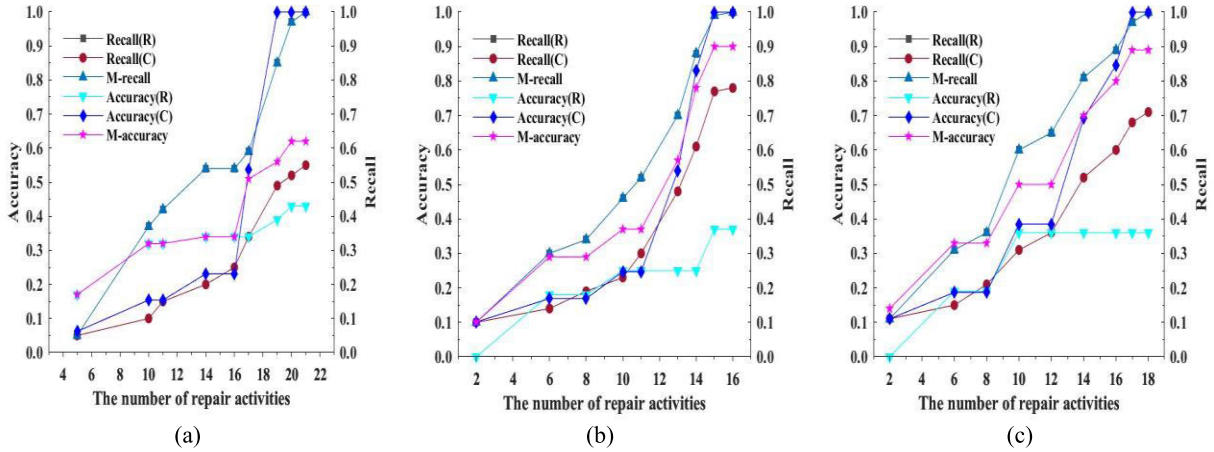

FIGURE 21. The number of repair activities, recall, and accuracy.

TABLE 6. Repair performance and range of application of various methods.

Ref	Fitness	Precision	Repair cost	Range of application
[34,35]	1	→	Minimum	General network structure
[36,37]	1	↑	Minimum	Alternative network structure
[38]	1	→	Minimum	General network structure
[17]	1	↑	Minimum	General network structure
[40]	《1	↑	Minimum	General network structure
This work	1	$\uparrow \geq \uparrow_{[17]}$	Minimum	General network structure

patterns, the hide operation was analyzed from different perspectives [33]. The two consistency checks were combined in this paper. We refined the reachable activity structure, and identified unfitted behavioral patterns and model-specific behavioral patterns through the conformance check of behavioral relationships. The optimal alignment and configured replay graph produced by the unfitted behavioral patterns were used to detect the complete information on deviation and configurable behavior, respectively. The optimal alignment was applied to the unfitted behavioral pattern with the iterative insert deviation to detect it. Otherwise, the unfitted behavioral pattern required the configured replay graph to observe the configurable behavior.

B. MODEL REPAIR

The goal of model repair is to enable the event log to replay on the repaired process model and maintain the unique behavior of the original model as much as possible. The pseudo-Boolean constraint was used to deal with multi-objective problems. The approach achieves the maximum replay of the model at the minimum cost [34]. Unnecessary redundant behaviors in the process model are avoided as much as possible by manual repair compared with automatic operations [35]. They improve fitness reasonably while reducing the cost of repair as much as possible. The defined extension alignment was added to the reachable marks of the process model, and deviations in the model were calculated and repaired [36]. The location of repair of the model was determined by Petri net and the transformation relationship between the alternative structures [37]. Various measure criteria were considered for the repair process model, but the above two methods were the only ones applied using the alternative structure. According to six algorithms, suggestions for repair were obtained under the given constraints, and a balance between fitness and computation was struck [38]. Each deviant element was repaired using a self-loop insert/skip according to the recommendations for repair, often at the expense of precision to improve fitness. The directly following non-iterative insert deviations in the concurrency behavior inclusion pattern were used to form a substructure to improve precision, but this is limited in terms of application [17]. In this paper, the directly following non-iterative insert deviations were reconstructed into a substructure according to the behavioral relationships detected by the optimal alignment applied to unfitted behavioral pattern with an iterative insert deviation.

C. CONFIGURATION OPTIMIZATION

Configuration optimization is compatible with the variables generated and used in the process of change based on the reference model, with common characteristics under the given constraints. It processes all configurable variables through adding or hiding the selection substructures of the control flow [39]. Configurable fragments are constructed from the empirical behavior recorded in the event log to produce a process model that can satisfy both the common behavior and the customized behavior [40]. In this paper, this configuration

optimization technique was introduced to model repair. The fitted behavior between the event log and the process model was regarded as a common parameter while the configurable behavior caused by unfitted behavior was regarded as a variable [41]. All iterative insert deviations cannot be replayed by configuration, which affects the improvement of the fitness [42]. Configurable behavior in the replay was optimized through the configuration operation when no iterative insert deviation occurred in the unfitted behavioral pattern in this paper. Through the configuration operation, the non-iterative deviation in the event log could be accurately replayed. Thus, fitness and precision were completely improved when the unfitted behavioral pattern was configured without iterative insert deviation. The features of existing four model repair methods and the proposed method are summarized in table 6. The deviations are inserted on process model using self-loops or skipped from process model by invisible transitions, and used the fitness and precision repaired by this method as a reference value [36]. $\rightarrow$, $\uparrow$, and $\downarrow$ represent unchanged, increase, and decrease after repair of the current method compared with the reference precision, respectively.

VII. CONCLUSION

This paper proposed a method to repair behavior that cannot be replayed in case of an uncertain insert deviation that answered the following two questions: i) how can precision be improved in cases of an unfitted behavioral pattern with an iterative insert deviation? ii) How can precision be improved when there is no iteration insert deviation in the unfitted behavioral pattern? The solution to the above problems was divided into the following parts:

1) The corresponding unfitted behavioral patterns were identified by refining the reachable activity structures of the event log and the process model. The optimal alignment and configured replay graph were produced to detect complete information on the deviation and the configurable behavior, respectively.
2) The process model was repaired by switching between operations based on the results of checking for iterative insert deviations in the given pattern. If they were present, the directly following non-iterative insert deviations were constructed into a substructure according to the behavioral relationship. The insert deviation or substructure was repaired using a self-loop insert. By contrast, the process was repaired using a conflict substructure combined with an invisible transition and configurable behavior. Therefore, the behavioral pattern with iterative insert deviation improves the precision by reducing the size of the event log and maintaining the cost of accurately repairable deviations. Otherwise, the fitness and precision can be completely repaired by configuration operation.

REFERENCES

[1] V. Huser, "Process mining: Discovery, conformance and enhancement of business processes," *J. Biomed. Inform.*, vol. 45, pp. 1018–1019, Jun. 2012.

[2] W. van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: Discovering process models from event logs," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 9, pp. 1128–1142, Sep. 2004.

[3] W. M. P. V. Der Aalst, "Distributed process discovery and conformance checking," in *Proc. Int. Conf. Fundam. Approaches Softw. Eng.*, 2012, pp. 1–25.

[4] E. Asare, L. Wang, and X. Fang, "Conformance checking: Workflow of hospitals and workflow of open-source EMRs," *IEEE Access*, vol. 8, pp. 139546–139566, 2020.

[5] D. Fahland and W. M. P. V. Der Aalst, "Repairing process models to reflect reality," in *Proc. Conf. Bus. process Manage.*, 2012, pp. 229–245.

[6] M. Weidlich, A. Polyvyanyy, N. Desai, J. Mendling, and M. Weske, "Process compliance analysis based on behavioural profiles," *Inf. Syst.*, vol. 36, no. 7, pp. 1009–1025, Nov. 2011.

[7] A. Rozinat and W. M. P. van der Aalst, "Conformance checking of processes based on monitoring real behavior," *Inf. Syst.*, vol. 33, no. 1, pp. 64–95, Mar. 2008.

[8] S. K. L. M. vanden Broucke, J. De Weerdt, J. Vanthienen, and B. Baesens, "Determining process model precision and generalization with weighted artificial negative events," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 8, pp. 1877–1889, Aug. 2014.

[9] A. Adriansyah and B. F. Van Dongen, "Cost-based conformance checking using the a algorithm," *BPM Center Report*, vol. 1111, pp. 1–14, May 2011.

[10] A. A. Adriansyah, B. F. Van Dongen, and W. M. P. V. Der Aalst, "Conformance checking using cost-based fitness analysis," in *Proc. Conf. Enterprise Distrib. Object Comput.*, Aug. 2011, pp. 55–64.

[11] D. Fahland and W. M. P. van der Aalst, "Simplifying discovered process models in a controlled manner," *Inf. Syst.*, vol. 38, no. 4, pp. 585–605, Jun. 2013.

[12] Z. He, Y. Du, L. Qi, and H. Du, "A model repair approach based on Petri nets by constructing free-loop structures," *IEEE Access*, vol. 7, pp. 24214–24230, 2019.

[13] W. van der Aalst, A. Adriansyah, and B. van Dongen, "Replaying history on process models for conformance checking and performance analysis," *WIREs Data Mining Knowl. Discovery*, vol. 2, no. 2, pp. 182–192, Mar. 2012.

[14] F. Mannhardt, M. de Leoni, H. A. Reijers, and W. M. P. van der Aalst, "Balanced multi-perspective checking of process conformance," *Computing*, vol. 98, no. 4, pp. 407–437, Apr. 2016.

[15] N. Macedo, T. Jorge, and A. Cunha, "A feature-based classification of model repair approaches," *IEEE Trans. Softw. Eng.*, vol. 43, no. 7, pp. 615–640, Jul. 2017.

[16] A. Adriansyah, J. Munoz-Gama, J. Carmona, B. F. van Dongen, and W. M. P. van der Aalst, "Measuring precision of modeled behavior," *Inf. Syst. E-Bus. Manage.*, vol. 13, no. 1, pp. 37–67, Feb. 2015.

[17] D. Fahland and W. M. P. van der Aalst, "Model repair—Aligning process models to reality," *Inf. Syst.*, vol. 47, pp. 220–243, Jan. 2015.

[18] M. De Leoni, W. W. V. Der Aalst, and B. F. Van Dongen, "Data-and resource-aware conformance checking of business processes," in *Proc. Int. Conf. Bus. Inf. Syst.*, 2012, pp. 48–59.

[19] M. Asadi, B. Mohabbati, G. Gröner, and D. Gasevic, "Development and validation of customized process models," *J. Syst. Softw.*, vol. 96, pp. 73–92, Oct. 2014.

[20] R. Conforti, M. L. Rosa, and A. H. M. T. Hofstede, "Filtering out infrequent behavior from business process event logs," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 2, pp. 300–314, Feb. 2017.

[21] X. Fang, R. Cao, X. Liu, and L. Wang, "A method of mining hidden transition of business process based on region," *IEEE Access*, vol. 6, pp. 25543–25550, 2018.

[22] J. Vanhatalo, H. Volzer, and J. Koehler, "The refined process structure tree," *Data Knowl. Eng.*, vol. 68, no. 9, pp. 793–818, 2009.

[23] T. Calders, C. Guenther, M. Pechenizkiy, and A. Rozinat, "Using minimum description length for process mining," in *Proc. SAC*, 2009, pp. 1451–1455.

[24] A. Armascervantes, P. Baldan, M. Dumas, and L. Garciabanuelos, "Diagnosing behavioral differences between business process models," *Inf. Syst.*, vol. 56, pp. 304–325, May 2016.

[25] N. Kleiner, "Delta analysis with workflow logs: Aligning business process prescriptions and their reality," *Requirement Eng.*, vol. 10, no. 3, pp. 212–222, Aug. 2005.

[26] M. La Rosa, W. M. P. V. Der Aalst, M. Dumas, and F. Milani, "Business process variability modeling: A survey," *ACM Comput. Surv.*, vol. 50, pp. 1–45, May 2017.

[27] J. C. A. M. Buijs, B. F. van Dongen, and W. M. P. van der Aalst, "Mining configurable process models from collections of event logs," in *Business Process Management*. Berlin, Germany: Springer, 2013, pp. 33–48.
[28] L. Garcia-Banuelos, N. R. T. P. van Beest, M. Dumas, M. L. Rosa, and W. Mertens, "Complete and interpretable conformance checking of business processes," *IEEE Trans. Softw. Eng.*, vol. 44, no. 3, pp. 262–290, Mar. 2018.
[29] M. de Leoni, F. M. Maggi, and W. M. P. van der Aalst, "An alignment-based framework to check the conformance of declarative process models and to preprocess event-log data," *Inf. Syst.*, vol. 47, pp. 258–277, Jan. 2015.
[30] T. Nguyen, A. Colman, and J. Han, "Modeling and managing variability in process-based service compositions," in *Proc. Int. Conf. Service Oriented Comput.*, 2011, pp. 404–420.
[31] J. C. A. M. Buijs, M. L. Rosa, H. A. Reijers, B. F. V. Dongen, and W. M. P. V. D. Aalst, "Improving business process models using observed behavior," in *Proc. Int. Symp. Data-Driven Process Discovery Anal.*, 2012, pp. 44–59.
[32] W. Song, X. Xia, H.-A. Jacobsen, P. Zhang, and H. Hu, "Efficient alignment between event logs and process models," *IEEE Trans. Services Comput.*, vol. 10, no. 1, pp. 136–149, Jan. 2017.
[33] N. L. Garca-Ba, B. N. Van, M. Dumas, and M. L. Rosa, "Business process conformance checking based on event structures," in *Proc. 27th Nordic Workshop Program. Theory*, vol. 2015, Art. no. 3.
[34] C. Di Francescomarino, R. Tiella, C. Ghidini, and P. Tonella, *A Multi-objective Approach to Business Process Repair*. Berlin, Germany: Springer, 2014, pp. 32–46.
[35] A. A. Cervantes, N. R. T. P. V. Beest, M. L. Rosa, M. Dumas, and L. García-Bauelos, "Interactive and incremental business process model repair," in *Proc. Int. Conf.*, 2017, pp. 53–74.
[36] H. Qi, Y. Du, L. Qi, and L. Wang, "An approach to repair Petri net-based process models with choice structures," *Enterprise Inf. Syst.*, vol. 12, nos. 8–9, pp. 1149–1179, Oct. 2018.
[37] X. Zhang, Y. Du, L. Qi, and H. Sun, "Repairing process models containing choice structures via logic Petri nets," *IEEE Access*, vol. 6, pp. 53796–53810, 2018.
[38] A. Polyvyanyy, W. M. P. V. D. Aalst, A. H. M. T. Hofstede, and M. T. Wynn, "Impact-driven process model repair," *ACM Trans. Softw. Eng. Methodol.*, vol. 25, no. 4, pp. 1–60, May 2017.
[39] N. Assy, W. Gaaloul, and B. Defude, "Mining configurable process fragments for business," *Process Des.*, vol. 8463, pp. 209–224, May 2014.
[40] W. M. P. van der Aalst, A. Dreiling, F. Gottschalk, M. Rosemann, and M. H. Jansen-Vullers, "Configurable process models as a basis for reference modeling," in *Proc. Int. Conf. Bus. Process Manage.*, vol. 3812, 2006, pp. 512–518.
[41] C. Li, M. Reichert, and A. Wombacher, "Mining business process variants: Challenges, scenarios, algorithms," *Data Knowl. Eng.*, vol. 70, pp. 409–434, May 2011.
[42] G. Gröner, F. Silva Parreiras, and D. Gaáevi, "Modeling and validation of business process families," *Inf. Syst.*, vol. 38, no. 5, pp. 709–726, Jul. 2013.

LIWEN ZHANG is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, Anhui University of Science and Technology, China. Her current areas of research include Petri net, process mining, model repair, and configuration optimization.

XIANWEN FANG received the M.A. degree from the Shandong University of Science and Technology, China, in 2004, and the Ph.D. degree from the Key Laboratory of Service Computing, Tongji University, in 2011. He is currently a Professor with the Department of Computer Science and Engineering, Anhui University of Science and Technology, China. His research interests include Petri net, trustworthy software, and Web services.

CHIFENG SHAO is currently pursuing the master's degree with the Department of Computer Science and Engineering, Anhui University of Science and Technology, China. His current areas of research include Petri net and process mining.

LILI WANG received the M.A. degree from the Shandong University of Science and Technology, China, in 2007. She is currently an Associate Professor with the Department of Computer Science and Engineering, Anhui University of Science and Technology, China. Her current areas of research include Petri net and formal verification of software.