Hindawi
Scientific Programming
Volume 2021, Article ID 8874316, 17 pages
https://doi.org/10.1155/2021/8874316

Research Article

An Optimization Approach for Mining of Process Models with Infrequent Behaviors Integrating Data Flow and Control Flow

Li-li Wang,[1,2] Xian-wen Fang ,[1] Esther Asare,[1] and Fang Huan[1]

[1]*College of Mathematics and Big Data, Anhui University of Science and Technology, Huainan 232000, China*
[2]*The Key Laboratory of Embedded System and Service Computing Ministry of Education (Tongji University), Shanghai 201804, China*

Correspondence should be addressed to Xian-wen Fang; xwfang@aust.edu.cn

Received 30 May 2020; Revised 30 July 2020; Accepted 15 April 2021; Published 3 May 2021

Academic Editor: Mu-Chen Chen

Infrequent behaviors of business process refer to behaviors that occur in very exceptional cases, and their occurrence frequency is low as their required conditions are rarely fulfilled. Hence, a strong coupling relationship between infrequent behavior and data flow exists. Furthermore, some infrequent behaviors may reveal very important information about the process. Thus, not all infrequent behaviors should be disregarded as noise, and identifying infrequent but correct behaviors in the event log is vital to process mining from the perspective of data flow. Existing process mining approaches construct a process model from frequent behaviors in the event log, mostly concentrating on control flow only, without considering infrequent behavior and data flow information. In this paper, we focus on data flow to extract infrequent but correct behaviors from logs. For an infrequent trace, frequent patterns and interactive behavior profiles are combined to find out which part of the behavior in the trace occurs in low frequency. And, conditional dependency probability is used to analyze the influence strength of the data flow information on infrequent behavior. An approach for identifying effective infrequent behaviors based on the frequent pattern under data awareness is proposed correspondingly. Subsequently, an optimization approach for mining of process models with infrequent behaviors integrating data flow and control flow is also presented. The experiments on synthetic and real-life event logs show that the proposed approach can distinguish effective infrequent behaviors from noise compared with others. The proposed approaches greatly improve the fitness of the mined process model without significantly decreasing its precision.

1. Introduction

The purpose of process mining is to extract useful knowledge from event logs recorded by IT systems of enterprises to discover, monitor, and enhance the actual business process [1, 2]. One of the important research areas is process discovery, which automatically infers process models from event logs. The goal of process discovery is to find the "best" process model given a record of the real executions as much as possible. The four important metrics for measuring the "best" model are fitness, precision, generalization, and simplicity [3].

Unfortunately, real-life event logs often contain both noise and infrequent behavior [2, 4, 5]. In general, noise refers to behavior that does not conform to a process specification and/or correct execution, such as traces of recorded incomplete process behaviors, recording errors, and error execution of process [5]. However, effective infrequent behavior is considered to be a possible execution behavior in very exceptional cases, such as fraudulent behavior in insurance [6], risk problems in system operations [7], and escape problems of spacecraft systems. Some infrequent behaviors may be important behaviors that cannot be discarded in system operation. Early process discovery algorithms [8–13] have assumed that event logs accurately record system behavior and apparently have significant limitations in real life. Most of the recent discovery algorithms [14–23] support noise filtering but ignore infrequent behaviors of business processes. Very few discovery algorithms [24–27] consider infrequent behaviors, whereas they

still regarded infrequent behaviors as noises. This may lead to some important information to be discarded. As a result, the derived models have difficulty in accurately describing the real behavior of systems. Therefore, one important challenge in process discovery is to distinguish infrequent behavior from noise in event logs.

There are few approaches related to the research of infrequent behavior, most of which focus on the control-flow perspective. Existing approaches determine whether the behavior is infrequent or frequent only by considering the frequency of activities or directly-follows relations. However, for the infrequent behavior, they rarely analyze whether it has a relationship with data flow and directly remove it as noise. However, in real system operation, some execution paths may be taken by contextual data information, such as available resources, execution time, and execution status. As the required conditions (that is, specific data information) are rarely fulfilled, some paths are executed infrequently. Therefore, these infrequent behaviors are caused by their special required conditions. Once these conditions are fulfilled, the corresponding infrequent behavior will inevitably occur. We can say these infrequent behaviors as effective infrequent behaviors or correct infrequent behaviors. For instance, an airbag deploying in a car requires a suitable speed and angle of impact. Theoretically, the airbag can only be opened when the impact on a fixed object is within 60° in front of the vehicle, and the car speed is higher than 30 km/h. Compared with normal driving activities, the frequency of airbag deployment behavior is lower. Obviously, there is a coupling relationship between these infrequent behaviors and the data information of the event. Moreover, it is an important behavior for system operation. Therefore, existing approaches that filter low-frequency behavior based on control flow and treat it as noise are not appropriate. Identifying these effective infrequent behaviors from the perspective of data flow and integrating control flow and data flow information in process discovery play an important role in process model optimization, business process improvement, resource allocation adjustment, and so on.

This paper analyzes the coupling relationship between infrequent behavior and data flow. It quantifies the influence strength of data information on behavioral dependencies between events, which provides a reliable basis for the identification of effective infrequent behavior. We conduct a series of experiments to compare our approach to existing approaches on synthetic and real-life event logs and discuss the result. The experimental result indicates that the proposed approach can identify more infrequent but useful behaviors than the state-of-the-art mining technique and greatly increase the fitness of the process model without significantly decreasing the precision and, indeed, optimizing the process model. The main contributions of the paper are as follows:

(1) An analysis approach based on frequent patterns and interactive behavior profiles is proposed to identify which parts of the trace are infrequent

(2) To quantify the strong influence of data information on the behavioral dependence between activities, a conditional dependence probability measurement approach is introduced

(3) An effective infrequent behavior recognition approach based on frequent patterns under data awareness is presented along with an optimization approach for mining of process models with infrequent behaviors integrating data flow and control flow

The remainder of this paper is structured as follows. Section 2 discusses the related work. Section 3 introduces the problem with an example. Section 4 presents the notations and the required concepts. Section 5 proposes an effective infrequent behavior recognition approach based on frequent patterns under data awareness. After that, an optimization approach for mining of the process model with infrequent behaviors that integrates data flow and control flow is also given. Section 6 evaluates how well the proposed approach works on synthetic and real-life event data. Finally, Section 7 concludes the paper and discusses future work.

2. Related Work

Many researchers have proposed a range of process mining algorithms. However, there exist many problems in the process mining algorithm, such as short loops [28], indirect dependency relationships [29], duplicated transitions, invisible transitions, noises, and infrequent behaviors [30]. Some early mining algorithms, such as the α-algorithm [8] and its derived improved algorithm [9, 10], the ILP mining algorithm [11], the inductive miner algorithm [12], and the domain-based mining algorithm [13], disregarded noises in the event log. Clearly, they have great limitations in real life. Most of the recent mining algorithms support noise filtering [14–23]. The first discovery algorithm handling noise was heuristics miner [14]. Heuristics miner considers the frequencies of the basic ordering relations during the computation of the strength of causal relations. The true dependency between two events (such as concurrency, exclusion, and causality) is determined by the strength of the causal relations. Its derived algorithms have also been proposed [15, 16]. Existing noise-filtering approaches are based on frequencies [14–18], machine-learning techniques [19, 20], genetic algorithms [21], or probabilistic models [22, 23]. All of those approaches focus on the control-flow perspective when filtering noise without considering the data flow information and exclude infrequent but useful behaviors. The literature [31–33] specifically studied the noise processing approach in event logs but still did not address infrequent behaviors.

Recently, the literature on infrequent behavior has been very scarce, and it has mainly focused on the control-flow perspective [24–27]. In terms of control flow, the literature [24] proposed the *WoMine-i* algorithm, which retrieves infrequent behavior patterns from a process model, including structures with sequences, selections, parallels, and loops. However, in general, we do not have a reference model in real life, only have logs recorded reality. Hence, it is difficult for this approach to improve the

quality of the discovered model. In [25], the inductive miner infrequent algorithm was proposed, which adds infrequent behavior filters to all steps of the IM algorithm, such that infrequent behavior is filtered by adopting an eventually-follows graph. In [26], a minimum anomaly-free automaton (AFA) based on the whole event log and a given threshold was constructed. Subsequently, all events that did not fit the AFA were removed from the filtered event log, which led to the removal of individual events rather than entire traces from the log. However, this technique cannot detect some typical anomalies, such as incomplete traces. The approaches in [25, 26] filter infrequent behaviors based on the frequency of directly-follows relation between the activity pairs only from the perspective of the control flow and neglect the dependence between some infrequent behaviors and data flow. In [27], the authors proposed a generic noise-filtering approach suitable for any arbitrary process discovery algorithms. The approach uses the conditional occurrence probability to calculate the likelihood of the occurrence of an activity following a subsequence. The disadvantage of this approach is that the log is interpreted as a sequence, whereas the structural information is not considered, such as concurrent and loop structures. For concurrent and loop structures, the same structure will correspond to multiple or even infinitely different subsequences. Additionally, distinguishing between noise and infrequent behavior is ignored, and they are directly deleted as noise. In terms of data flow, in [34], the data-aware heuristic miner (DHM) was proposed, which combines data flow and control flow. A classification technique is used to find the data attributes between activities, which can reveal conditional infrequent behavior from the event log to distinguish infrequent behavior from noise effectively. There are two limitations to this approach. First, according to the condition of data dependence, only the dependency strength of two different directly-follows relations between activity a and activity b is computed, but the probability of activity b following activity a directly compared to other activities is not considered. Second, only condition directly-follows dependencies between activities are discovered, and the conditional dependencies of the more complex patterns cannot be discovered. Recent work on declarative process discovery [35–37] considered the data perspective. Declarative process models representing the discovery results from execution logs are given. In [37], the authors present an automated discovery of a declarative process model with data conditions. Clustering techniques in conjunction with a rule mining technique and redescription mining techniques are used to discover constraints between two activities, respectively. However, similar to association rule mining, only sets of rules or constraints rather than full process models are returned.

As a consequence, there exist three problems with the process discovery algorithm in recognition of infrequent behavior. First, most of them focus on the control-flow perspective to recognize infrequent behavior, whereas they ignore the coupling relationship between infrequent behavior and data conditions. Second, they directly remove all infrequent behaviors as noise when discovering the process model, which leads to infrequent but useful behaviors which are also excluded. Third, these approaches only consider the directly-follows dependency of activity pairs, while the dependencies of more complex patterns are ignored. For the reasons stated above, this paper proposes an effective infrequent behavior recognition approach based on frequent patterns under data awareness. Subsequently, an optimization approach for mining of process models with infrequent behaviors integrating data flow and control flow is provided.

3. Motivation

A business process of booking tickets in a train ticket reservation system is illustrated as an example. Here, only the business process of ordering the ticket is considered, and a series of business processes generated by refunds and changes are not considered. Assuming that only 1,000 records are extracted from the system log, the trace sequences and the frequency of their occurrence are shown in Table 1. The event names corresponding to the activities are shown in Table 2.

Assuming a frequent threshold is 100, the inductive miner-infrequent (IMi) algorithm [25] infers the initial process model after removing the infrequent traces $\sigma_7, \ldots, \sigma_{12}$, as shown in Figure 1.

In a real system, σ_8, σ_9, and σ_{12} are three effective infrequent traces corresponding to situations in which the total number of contact names related to the logged-in user exceeds 15. The time interval between confirming the order and the payment success is more than 30 minutes, the user has an unpaid order, and the waiting time for payment does not timeout. In particular, they often occur infrequently because the corresponding conditions are rarely fulfilled. Obviously, these behaviors are infrequent but correct. However, the IMi algorithm disregards these as noise when constructing the process model so that the resulting process model cannot truly describe the actual operation of the system.

In trace σ_8, it is not difficult to find an infrequent activity K′, which has an indirect data-dependent relationship with the frequent activity F. In trace σ_9, there is a low-frequency activity pair TQ, which is caused by an indirect data dependency between R and T. The reason that trace σ_{12}occurs infrequently is the same as that for the trace σ_9. It is obvious that there is a coupling relation between these infrequent behaviors and the particular data dependency. To capture the infrequent and useful behavior, an effective infrequent behavior recognition approach based on the frequent pattern under data awareness is proposed in this paper according to indirect data dependency between events. Furthermore, an optimization process model integrating data flow and control flow is obtained by incorporating infrequent behavior in the resulting process model, which increases the fitness of the process model and more accurately captures the important behaviors of the system.

Table 1: Event log for booking tickets in the booking system.

Id	Frequency	Trace
σ_1	179	ACDFBKLMRUVRTVRSW
σ_2	162	DEDEDFABKJJLMNPQ
σ_3	108	DEDFABGABJKKLMRTVRSW
σ_4	111	CDFABGABKKLKJLMRUVRTVRUVRSW
σ_5	121	ABDFKKLMNONONPQ
σ_6	113	DFABJJKLMNORUVRSW
σ_7	80	DEDDFABGAB
σ_8	16	ABDFJKK′KLMRSW
σ_9	40	ABDFJLMRTQ
σ_{10}	10	DFABJLMRTW
σ_{11}	43	ABDFJJMLNRSW
σ_{12}	17	ABDEDFRSW

Table 2: The event name corresponding to the activity.

Activity name	Event
A	Search
B	Booking
C	Register
D	Login
E	Failure
F	Success
G	Return
J	Select passenger
K	Insert passenger
K′	Remove passenger
L	Submit
M	Confirm an order
N	Cancel an order
O	Step back
P	Confirm cancel
Q	Successful cancel
R	Online payment
S	Successful payment
T	No payment
U	Payment failure
V	Continue to payment
W	Order completion

4. Preliminaries

This section gives basic definitions of several terms used in this paper. The events in the log represent activities, the event log is a collection of traces, and the same trace may appear multiple times in the event log, with each trace corresponding to the execution of a process. The event log typically stores considerable additional information about the event, such as the active execution resource (such as people or devices) and the timestamp of the event execution.

Definition 1 (process model [38]). A process model is a quadruple $N = (P, T, F, C)$ with

(1) P and T as a nonempty set of place and transition, respectively

(2) $P \neq \varphi$, $T \neq \varphi$, and $P \cap T = \varphi$

(3) $F \subseteq (P \times T) \cup (T \times P)$ as the flow relation of N

(4) $\mathrm{dom}(F) \cup \mathrm{cod}(F) = P \cup T$, with
$\mathrm{dom}(F) = \{x \in P \cup T | \exists y \in P \cup T, (x, y) \in F\}$
$\mathrm{cod}(F) = \{x \in P \cup T | \exists y \in P \cup T, (y, x) \in F\}$

(5) $C = \{o\ d, or, pl, cy\}$ as the structure type of the process model for sequence, selection, parallel, and loop

Definition 2 (weak order (log) [39]). Let L be the event log. Let A_L be an activity set of L. The weak order relation $\succ_L \subseteq (A_L \times A_L)$ contains all pairs (x, y) such that there exists a trace $L_p = n_1, \ldots, n_m$ in L with $j, k \in \{1, \ldots, m\}$ and $j < k \leq m$ for which $n_j = x$ and $n_k = y$ hold.

Definition 3 (behavioral profile (log) [40]). Let L be the event log. Let A_L be an activity set of L. A pair $(x, y) \in (A_L \times A_L)$ is in at most one of the following relations:

(1) The strict order relation $\longrightarrow_L$, iff $x \succ_L y$ and $y \succ_L x$

(2) The exclusiveness relation $+_L$, iff and $x \succ_L y$ and $y \succ_L x$

(3) The interleaving order relation $||_L$, iff $x \succ_L y$ and $y \succ_L x$

The set $B_L = \left\{ \longrightarrow_L, +_L, \|_L \right\}$ is the behavioral profile of L.

Note that we say that a pair (x, y) is in reverse strict order, denoted by $y \leftarrow_L x$ if and only if $x \longrightarrow_L y$.

Definition 3 indicates that if any trace in the log does not contain both the activity x and the activity y, then $x +_L y$. If there are two different traces such that $x \succ_L y$ and $y \succ_L x$ hold, or there is a trace such that $x \succ_L y$ and $y \succ_L x$ both hold, then $x \|_L y$. If there is a trace for which $x \succ_L y$ holds and there is no other trace such that $y \succ_L x$, then $x \longrightarrow_L y$.

Definition 4. (causal behavioral profile (log) [40]). Let L be the event log. Let A_L be an activity set of L. $B_L = \left\{ \longrightarrow_L, +_L, \|_L \right\}$ is the behavioral profile of L.

(1) A pair $(x, y) \in \in (A_L \times A_L)$ is in the co-occurrence relation $\gg$, iff $\forall L_P \in L$ implies that $x \in L_P \Rightarrow y \in L_P$

(2) The set $CB_{\mathrm{L}} = \left\{ \longrightarrow_L, +_L, \|_L, > >_L \right\}$ is the causal behavioral profile of L

Clearly, the co-occurrence relation compensates for the option of the strict order relation. A causality holds between two activities x and y if they are in strict order $x \longrightarrow_L y$ and for any trace in the log must contain the activity y as long as it contains the activity x.

5. An Optimization Approach for Mining of Process Models with Infrequent Behaviors Integrating Data Flow and Control Flow

This section describes an approach for identifying effective infrequent behaviors from the perspective of data dependency and gives an algorithm to reconstruct optimized process models integrating data flow and control flow by incorporating effective infrequent behavior. Section 5.1 presents some relevant definitions and an algorithm for an

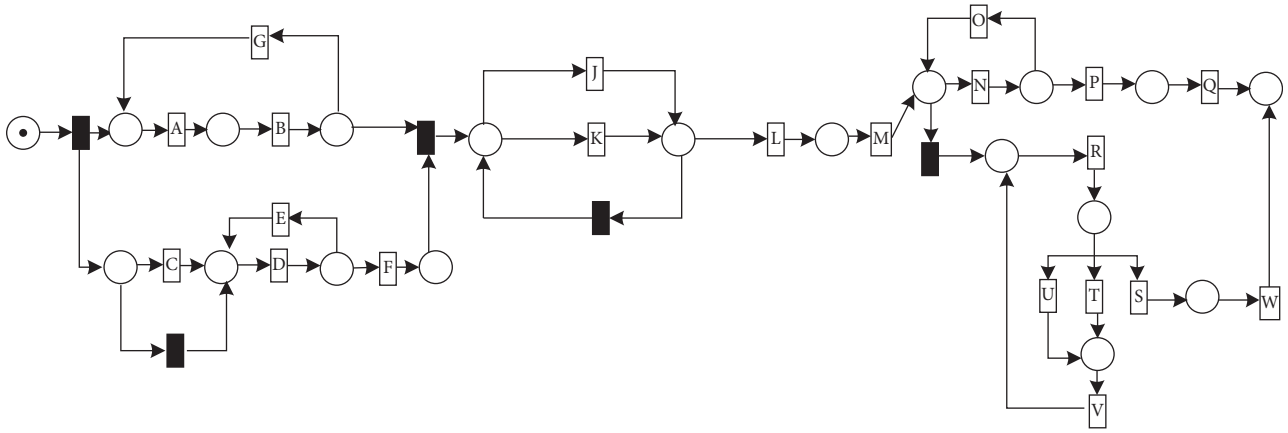

Figure 1: Initial process model M_1 of scheduled tickets.

effective infrequent behavior recognition approach based on frequent patterns under data awareness. Section 5.2 gives an optimization approach for mining of the process model with infrequent behaviors integrating data flow and control flow. The research framework of the proposed approach is shown in Figure 2.

5.1. An Effective Infrequent Behavior Recognition Approach Based on Frequent Patterns under Data Awareness. In this section, first (Section 5.1.1), some definitions related to the proposed approach are introduced, such as pattern, subsequence matching a pattern, interaction behavioral profile, and conditional dependency probability. Then, Section 5.1.2 elaborates on how to identify effective infrequent behaviors by using frequent patterns, interactive behavior profiles, and conditional dependency probabilities.

5.1.1. The Relation between Infrequent Behavior and Data Dependency. Prior to presenting the filtering approach, we present some basic notations used throughout the paper. Let A denote the set of all possible activities and let A^* denote a set of finite sequences over A. A finite sequence σ of length n over A is a function: σ: $\{1, 2, \ldots, n\} \longrightarrow A$, alternatively written as $\sigma = \langle a_1, a_2, \ldots, a_n \rangle$, where $a_i = \sigma[i]$ for $1 \le i \le n$. The empty sequence is written as ε. The concatenation of sequences σ and σ' is written as $\sigma \bullet \sigma'$. A sequence $\sigma\prime = \langle a_1', a_2', \ldots, a_k' \rangle$ is a subsequence of sequence σ if and only if we can write σ as $\sigma_1 \bullet \langle a_1', a_2', \ldots, a_k' \rangle \bullet \sigma_2$, where both σ_1 and σ_2 are allowed to be ε, i.e., σ is a subsequence of itself. The beginning activity of a finite trace σ is written as $\text{firstAct}(\sigma) = \sigma[1]$, and the end activity of a finite trace σ is written as $\text{lastAct}(\sigma) = \sigma[n]$ with $|\sigma| = n$. The set of all beginning activities and all ending activities in the event log L is written as startActs_L and endActs_L, respectively, where $\text{startActs}_L = \cup_{\sigma \in L} \text{firstAct}(\sigma)$ and $\text{endActs}_L = \cup_{\sigma \in L} \text{lastAct}(\sigma)$.

Considering the event logL_1, including five traces, where $\sigma_1 = \langle a, b, c, g \rangle^{10}$, $\sigma_2 = \langle a, c, b, g \rangle^5$, $\sigma_3 = \langle a, d, e \rangle^6$, $\sigma_4 = \langle a, d, e, f, d, e \rangle^8$, and $\sigma_5 = \langle a, d, e, f, d, e, f, d, e \rangle^4$ (the superscript of the trace indicates the number of times the trace appears in the log). Figure 3 shows the directly-follows graph $G(L_1)$ of log L_1. In the directly-follows graph, each node represents an activity, and an edge represents the directly-follows relationship between two activities in the trace. ⇨ indicates the start node of the log, double circles ⓖ indicate the end node of the log, a line with a double-sided arrow indicates two activities are in a concurrent relationship, and a line with a single-sided arrow indicates a directly-follows relationship.

According to Definition 3, we obtain the behavioral profile between activities in the log L_1 shown in Figure 4. Clearly, σ_1 and σ_2 are distinguishing traces, but in fact, they are the behavioral equivalent, as activity b and activity c are in an interleaving order relation. The same is true for traces σ_4 and σ_5. Therefore, traditionally treating the trace as a sequence is too imprecise. We consider that the sequences of equivalent behaviors are the same, even if their corresponding sequences are different. To analyze the frequency and correctness of subsequences included in a trace from the behavior perspective, we define a pattern that considers all types of structures—sequence, selection, concurrent, and loop.

Definition 5 (pattern). Given the event log$L$, let A_L be an activity set of L. Let $G(L)$ be a directly-follows graph of log L. All vertices of $G(L)$ are written as $V(G)$, and all edges of $G(L)$ are written as $E(G)$. When a connected subgraph satisfies the following two conditions, we call it a pattern of log L:

(1) $\forall a \in V\prime \Rightarrow a \in V(G)$, $\forall e \in E\prime \Rightarrow e \in E(G)$

(2) $\forall a, b \in V\prime$, $\forall x \in (A_L / V\prime)$, $CB_L(a, x) \Leftrightarrow CB_L(b, x)$

Definition 5 indicates that the pattern represents part of the behavior of the trace in the event log, and any activity in the pattern has the same behavioral profile relation with other activities that are not in this pattern. For convenience, the vertices of the pattern Patt are written as $V(\text{Patt})$, the edges of the pattern Patt are written as $E(\text{patt})$, and the pattern to which the activity a belongs is written as $\text{Patt}(a)$.

In the example provided in Figure 5, according to Definition 5, for $b, c \in V(\text{Patt 2})$, $d \in V(\text{Patt 3})$, and $d \notin V(\text{Patt 2})$, then $b +_L d \Rightarrow c +_L d$. For $a \in V(\text{Patt 1})$ and $a \notin V(\text{Patt 2})$, then $b \leftarrow_L a \Rightarrow c \leftarrow_L a$.

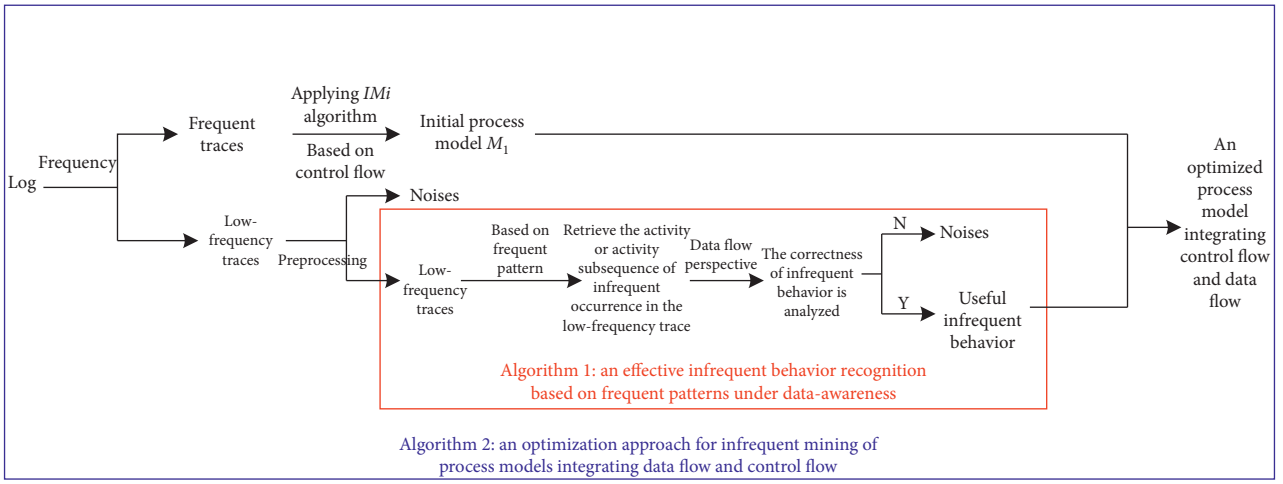

Figure 2: The research framework of the proposed approach.

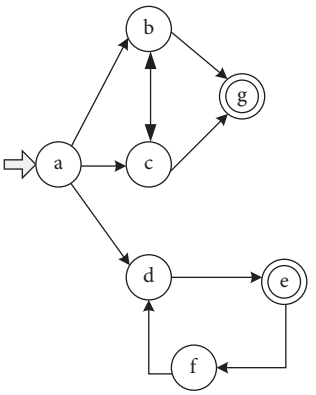

Figure 3: The directly-follows graph $G(L_1)$ of the log L_1.

	a	b	c	g	d	e	f
a		$\rightarrow_L$	$\rightarrow_L$	$\rightarrow_L$	$\rightarrow_L$	$\rightarrow_L$	$\rightarrow_L$
b	$\leftarrow_L$		$\|\|_L$	$\rightarrow$ L	$+_L$	$+_L$	$+_L$
c	$\leftarrow_L$	$\|\|_L$		$\rightarrow$ L	$+_L$	$+_L$	$+_L$
g	$\leftarrow_L$	$\leftarrow$ L	$\leftarrow$ L		$+_L$	$+_L$	$+_L$
d	$\leftarrow_L$	$+_L$	$+_L$	$+_L$		$\|\|_L$	$\|\|_L$
e	$\leftarrow_L$	$+_L$	$+_L$	$+_L$	$\|\|_L$		$\|\|_L$
f	$\leftarrow_L$	$+_L$	$+_L$	$+_L$	$\|\|_L$	$\|\|_L$	

Figure 4: Behavioral profile between activities in the log.

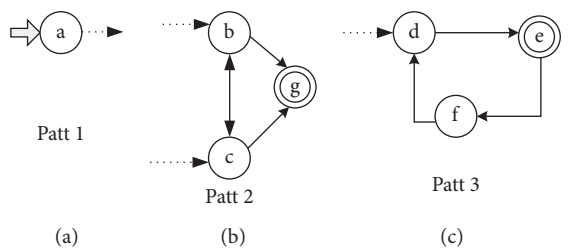

Figure 5: Three subpatterns of $G(L)$. (a) The subpattern Patt 1. (b) The subpattern Patt 2. (c) The subpattern Patt 3.

In expressing the interaction behavior between patterns better, the concept of an interactive behavioral profile is introduced as follows.

Definition 6 (interactive successor relationship and interactive input (or output) arc). Given the event log L, let A_L be an activity set of L and Patt be one of the patterns of the log L. For$\forall a, b \in A_L$, we denote $a \preccurlyeq_I b$ as an interactive successor relationship between the activity a and the activity b, if $\text{Patt}(a) \neq \text{Patt}(b)$ and $\exists \sigma \in L$ such that $\sigma[i] = a\sigma[i+1] = b$ (where $1 \leq i < |\sigma|$). And we say that the activity b has an interactive input arc and the activity a has an interactive output arc, respectively.

Definition 6 indicates that when the activity a and the activity b belong to different patterns and a trace exists in forms of $\langle \cdots, a, b, \cdots \rangle$, there exists an interactive successor relationship between a and b. For instance, in Figure 5, there are two interactive successor relationships between patterns Patt 1 and Patt 2: $a \preccurlyeq_I b$and $a \preccurlyeq_I c$.

An activity a is referred to as the entry node of pattern Patt if $a \in \text{startActs}_L$ or a has an interactive input arc. Similarly, an activity a is referred to as the exit node of the pattern if $a \in \text{endActs}_L$ or a has an interactive output arc. As a pattern may have multiple entry nodes, or multiple exit nodes, the set of entry nodes of pattern Patt is written as entry (Patt), and the set of exit nodes of pattern Patt is written as exit (Patt).

In a pattern, $\cdots\!\blacktriangleright\!(\text{a})$ indicates the interactive input arc of node a and $(\text{a})\!\cdots\!\blacktriangleright$ indicates the interactive output arc of node a.

According to Definition 5, the directly-follows graph of the aforementioned log L_1 can be divided into three highly cohesive low-coupling subpatterns, as shown in Figures 5(a)–5(c).

Definition 7 (subsequence matching a pattern). Let L be an event log over a set of activities A. Let pattern Patt be a subpattern of log L. A subsequence $\sigma\prime \in A^*$ in the trace is said to match a pattern Patt, denoted as $\sigma\prime \sqsubseteq \text{Patt}$, when $\sigma\prime = \text{ActSeq}(\text{path}(s, e))$ with $s \in \text{entry}(\text{Patt})$ and $e \in \text{exit}(\text{Patt})$ holds, wherepath (s, e) denotes a path from node s to node e, and ActSeq (path (s, e)) denotes the sequence of activities that consists of all nodes on the path from node s to node e.

Definition 7 illustrates that a subsequence is considered to match the pattern Patt when it corresponds to a substring consisting of all nodes on the path from the entry node to the exit node in the pattern Patt. The pattern that matches a subsequence σ is denoted as $\text{Patt}(\sigma)$.

For example, for the subsequences a, defde belonging to the traces $\sigma_4 = \langle a, d, e, f, d, e \rangle$ in logL_1, they match patterns Patt 1 and Patt 3, respectively. For different subsequences matching the same pattern, we consider them to be behavioral equivalent, i.e., although σ_1 and σ_2 are two different sequences, they are considered to be behavioral equivalent. σ_4 and σ_5 do the same.

Definition 8 (interactive behavioral profile (pattern)). Given the event log$L$, let A_L be an activity set of L and $\preccurlyeq_I$ be an interactive successor relationship. The interactive behavioral profile is the 3-tuple $(\longrightarrow_I, \preccurlyeq_{I,+I})_L$ defined by

(i) $a \longrightarrow_I b$ iff $a \preccurlyeq_I b$ and $b \not\preccurlyeq_I a$

(ii) $a \|_I b$ iff $a \preccurlyeq_I b$ and $b \preccurlyeq_I a$

(iii) $a +_I b$ iff $a \not\preccurlyeq_I b$ and $b \not\preccurlyeq_I a$

Also, we say that a transition pair (b, a) is in reverse strict order of the interaction, denoted by $b \leftarrow_I a$ if and only if the transition pairs (a, b) satisfy the strict order of the interaction, i.e., $a \longrightarrow_I b$.

According to Definition 8, the interactive behavioral profile of patterns Patt 1, Patt 2, and Patt 3 is shown in Figure 6.

An infrequent trace occurs at low frequencies either because it contains low-frequency events or because it contains low-frequency subsequences, along with a large number of high-frequency subsequences. How can the frequency of occurrence of subsequences or activities be determined? To solve this problem, the activity frequency and pattern frequency are both given below.

Definition 9 (activity frequency [24]). Given the event log L, let A_L bean activity set of L. The frequency $\text{ActFreq}(a)$ of an activity $a \in A_L$ is defined as

$$\text{ActFreq}(a) = \frac{|[\sigma \in L | a \in \sigma]|}{|L|}. \quad (1)$$

Given a frequency threshold min ActFreq, an activity a is frequent iff $\text{ActFreq}(a) \geq \min \text{ActFreq}$.

A pattern can reflect the structural behavior relationship between activities. Since there exist multiple different subsequences with the same behavior corresponding to the same pattern, so it is more accurate to measure their frequency by using the frequency of patterns.

Definition 10 (pattern frequency). Let L be an event log over a set of activities A. Let pattern Patt be a subpattern of log L. The frequency $P\text{Freq}(\text{Patt})$of a pattern Patt is defined as

$$P\text{Freq}(\text{Patt}) = \frac{|[\sigma \in L | \exists \sigma\prime, \sigma\prime \sqsubseteq \text{Patt} \wedge \sigma = \cdots \sigma\prime \cdots]|}{|L|}. \quad (2)$$

Given frequency thresholds min PFreq, a pattern Patt is frequent, iff $P\text{Freq}(\text{Patt}) \geq \min P\text{Freq}$.

Theorem 1. *Let L be an event log over a set of activities A. Let pattern Patt be a subpattern of log L.Given a subsequence $\sigma\prime \in A^*$, σ' is frequent iff $\text{Patt}(\sigma\prime)$ is frequent.*

Proof. According to Definition 12, we can easily obtain this conclusion.

Frequency-based filtering techniques only consider direct dependencies between activity pairs, while the frequency of directly-follows relation between all activity pairs

	a	b	c	d	e	f
a		$\rightarrow_I$	$\rightarrow_I$	$\rightarrow_I$	$\rightarrow_I$	$\rightarrow_I$
b	$_I\leftarrow$			$+_I$	$+_I$	$+_I$
c	$_I\leftarrow$			$+_I$	$+_I$	$+_I$
d	$_I\leftarrow$	$+_I$	$+_I$			
e	$_I\leftarrow$	$+_I$	$+_I$			
f	$_I\leftarrow$	$+_I$	$+_I$			

Figure 6: The interactive behavioral profile of Patt 1, Patt 2, and Patt 3.

in some infrequent trace is frequent. For instance, <a, b> and <b, c> are frequent activity pairs in the log, but the subsequences<a, b, c> consisting of <a, b> and <b, c> may be low-frequency subsequences. Only using the direct dependency between activity pairs will not identify that <a, b, c> is an infrequency subsequence. Therefore, filtering infrequent behavior only from the frequency of occurrence of a single activity pair is too imprecise. In this case, computing the probability that a certain activity directly occurs after the occurrence of the subsequence at larger distances is necessary. Definitions 11 and 12 compute the number of conditional occurrence times and conditional dependency probability, respectively, of the activity directly following the subsequence, when the data dependency condition C exists between a subsequence and an activity. □

Definition 11 (conditional occurrence times). Given a subsequence $\sigma\prime$, an activity $a \in A$, and dependency conditions C, we write $\mathrm{COT}(\sigma\prime^{C,L} > a)$ to represent the conditional occurrence times of a subsequence $\sigma\prime$ with the latest attribute values x directly followed by an activity a under dependency condition C; we denote $\mathrm{COT}(\sigma\prime^{C,L} > a)$ as

$$\mathrm{COT}(\sigma\prime^{C,L} > a) = \left|\left\{\sigma \middle| \exists \sigma \in L, \sigma = \cdots \sigma\prime \bullet < a > \cdots\right\}\right| + \left|\left\{\tilde{\sigma} \middle| \exists \tilde{\sigma} \in L, \tilde{\sigma} = \cdots \tilde{\sigma}\prime \bullet < a > \cdots, \tilde{\sigma}\prime \text{ match the patt}(\sigma\prime)\right\}\right|, \quad (3)$$

where $\mathrm{patt}(\sigma\prime)$ represents a pattern to which a subsequence $\sigma\prime$ matches it.

For example, traces $\sigma_1$and σ_2 in the previously mentioned log L_1, let (1) the activity g follow directly after the subsequence ab with the latest attribute values x_1 in the trace ⟨abcg⟩, and (2) the activity g also follow directly after the subsequence ba with the latest attribute values x_1 in the trace ⟨acbg⟩. According to Definition 11, $\mathrm{COT}(bc^{C,L} > g) = \mathrm{COT}(cb^{C,L} > g) = 15$. Definition 11 considers the number of conditional occurrences of which more behavioral equivalence subsequences are directly followed by the same activity (such as concurrency or loops) under the same conditions.

Behavioral dependencies between activities in real-world systems may be affected by direct or indirect data dependency between activities. To capture the strength of behavioral dependence they cause, Definition 12 further gives the concept of conditional dependency probability between the subsequence and the activity based on the literature [20].

Definition 12 (conditional dependency probability). Let L be an event log over a set of activities A. Given a subsequence $\sigma\prime \in A^*$, an activity $a \in A$, and dependency conditions C, we write $\mathrm{CDP}(\sigma' \Rightarrow a)$ to represent a conditional dependency probability of the subsequence with the latest attribute value x directly followed by the activity a under dependency conditions C; we denote $\mathrm{CDP}(\sigma' \Rightarrow a)$ as

$$\mathrm{CDP}(\sigma\prime \Rightarrow a) = \begin{cases} \dfrac{\mathrm{COT}\left(\sigma'^{C,L} > a\right) - \sum_{\overline{a} \in A} \mathrm{COT}\left(\sigma'^{C,L} > \overline{a}\right)}{1 + \mathrm{COT}\left(\sigma'^{C,L} > a\right) + \sum_{\overline{a} \in A} \mathrm{COT}\left(\sigma'^{C,L} > \overline{a}\right)}, & \text{if occur Times}_L(\sigma') \neq 0, \\ 0, & \text{else}, \end{cases} \quad (4)$$

where $\overline{a}$ represents other activities except activity a and $\text{occur Times}_L(\sigma') = |\{\sigma | \sigma = \cdots \sigma' \cdots, \text{where } \sigma \in L\}| + |\{\sigma | \sigma = \ldots\ldots \sigma'' \ldots\ldots \wedge \sigma'' \text{ match the patt}(\sigma')\}|$.

Obviously, the value of $\mathrm{CDP}(\sigma' \Rightarrow a)$ is a real number in (–1, 1). When the dependency condition C has the latest attribute value x, the higher the value of $\mathrm{CDP}(\sigma\prime \Rightarrow a)$, the more likely the activity a directly follows the subsequence $\sigma\prime$. If an infrequent subsequence has a higher value of $\mathrm{CDP}(\sigma\prime \Rightarrow a)$, it can be judged to be a correct infrequent behavior.

For a given conditional dependency probability threshold θ_{dep}, $\sigma' \bullet a$ is considered to be a reasonable subsequence under the current data dependency iff $\mathrm{CDP}(\sigma' \Rightarrow a) \geq \theta_{\mathrm{dep}}$.

Definition 13 (special data dependency). Given an event log L, a subsequence σ, an activity $a, b \in A$, and dependency conditions C or C'. C or C' is regarded as special data dependency if the following two conditions are met:

$$\frac{\mathrm{COT}(\sigma^{C,L} > a)}{|L|} \geq \theta \text{ if } C_{\text{value}} = x,$$

$$\frac{\mathrm{COT}(\sigma^{C,L} > b)}{|L|} \ll \theta \text{ or } \frac{\mathrm{COT}(\sigma^{C',L} > b)}{|L|} \ll \theta \text{ if } C_{\text{value}} = y \text{ or } C'_{\text{value}} = z, \quad (5)$$

where θ is a frequent threshold, $|L|$ is the total number of traces in the log, and $C_{\text{value}} = x$ represents the dependency condition C has the latest attribute value x.

5.1.2. An Effective Infrequent Behavior Recognition Approach Based on Frequent Patterns under Data Awareness. Definition 12 in Section 5.1.1 quantifies the strength of data dependency on the behavioral dependency between the activity and the subsequence, which provides a basis for the identification of effective infrequent behaviors. This section provides an effective infrequent behavior recognition approach based on frequent patterns under data awareness.

For the frequent traces in the log, a number of patterns with high-cohesion low-coupling on behavior can be constructed by their directly-follows graph and behavioral profile. It is easy to determine that these subpatterns are frequent patterns. Since an infrequent trace often contains some frequent behavior in addition to infrequent behavior, there may be direct or indirect data dependency between them. To make full use of this dependency and accurately capture its impact on behavioral relationships, Algorithm 1, first, finds out which part of the behavior in the trace occurs in low frequency. Then, check whether there exists special data context information in the context of the infrequent behavior. If it exists, conditional dependency probability is used to analyze the influence strength of the data flow information on infrequent behavior. If its value is greater than a certain threshold, the infrequent behavior is considered to be an infrequent but correct behavior; otherwise, it is considered to be noise.

Step 1–Step 11 in Algorithm 1 analyze the validity of the infrequent trace in terms of the infrequent activity, where Step 1–Step 3 determine whether there is an infrequent activity in the trace and if it exists, Step 4–Step 11 determine the correctness of the infrequent activity occurrence from a data dependence perspective. Step 12–Step 13 analyze the validity of the infrequent trace in terms of the infrequent subsequence. Step 12 divides the trace σ into several subsequences according to the activity set in the frequent subpatterns. The divided subsequence is either a frequent subsequence or an infrequent subsequence. If the divided subsequence includes a smaller infrequent subsequence, Step 16–Step 21 analyze the correctness of the infrequent subsequence from the perspective of data dependence. If the divided subsequences are all legal subsequences, Step 21–Step 33 determine whether the interaction behavior between the subsequences is reasonable according to the interaction behavior profile of the frequent subpatterns. If it is unreasonable, the correctness of infrequent interaction between them is judged from the data dependency perspective.

5.2. An Optimization Approach for Mining of Process Models with Infrequent Behaviors Integrating Data Flow and Control Flow. Algorithm 1 analyzes the effectiveness of infrequent behavior based on direct or indirect data dependencies between events. On this basis, Algorithm 2 further gives an optimization approach for mining of the process model with infrequent behaviors integrating data flow and control flow. First, the initial process model based on the control flow is constructed from the frequent traces by using the *IMi* mining algorithm. Then, Algorithm 1 is used to identify all the effective infrequent behaviors in the event log. Finally, an optimization process model integrating data flow and control flow is further reconstructed by incorporating all the effective infrequent behaviors into the initial process model.

Step 1–Step 4 in Algorithm 2 preprocess the traces in the event log according to the occurrence frequency and divide the log into two sets, FilterLog andOutlier, where FilterLog represents all frequent traces and represents the infrequent traces that need to be analyzed. The initial process model is built by applying the *IMi* mining algorithm on the event log in Step 5. Incomplete traces that do not start or end normally are deleted and simultaneously added to the set Noise in Step 6–Step 11. Step 12–Step 14 use Algorithm 1to determine whether each trace in the set Outlier is an effective infrequent trace and further divide the trace into two subsets: effective infrequent trace set Infrequent and noise set Noise. An optimized process model M_2 of the fusion control flow and data flow is obtained by adding these infrequent behaviors in the set Infrequent to the initial process model M_1 in Step 15.

6. Evaluations and Results

In this section, we conducted controlled experiments on synthetic and real-life event logs to compare our approach to existing approaches and discuss the result in this section. First, we (in Section 6.1) illustrate the solution steps of the infrequent behavior identification approach proposed in this paper using the synthetic event log shown in Section 3 and then report on the number of infrequent behaviors correctly identified using our approach and other approaches. Then, in Section 6.2, we compare the proposed approach with other approaches to measure the quality of the process model discovered when different levels of infrequent behavior are injected into the real-life logs. These experiments are performed on an Intel i7-6500 processor and an 8 GB RAM (2.50).

6.1. Synthetic Dataset. In verifying the correctness of Algorithm 1, the event log given in Section 3 is taken as an example to elaborate on how to use Algorithm 1 to identify effective infrequent behavior. First, the causal behavioral profile is obtained according to frequency traces in the event log, as shown in Figure 7 (note that the subscript L of the behavioral profile is omitted here). Six maximal frequent patterns obtained according to the behavioral profile in Figure 7 are presented in Figure 8. The corresponding interactive behavior profiles between them are shown in Figure 9.

For the infrequent traces $\sigma_7, \sigma_8, \sigma_9, \sigma_{10}, \sigma_{11}$, and σ_{12}, since the end activity of trace σ_7 is an abnormal end activity, it is easy to determine that it is noise. In the event log, the attributes of some activities and their attribute values of infrequent traces $\sigma_8, \sigma_9, \sigma_{10}, \sigma_{11}$, and σ_{12} are shown in Table 3 to Table 7, respectively. The conditional dependence probabilities between certain activities and subsequences are computed according to Algorithm 1, as displayed in Table 8.

Input: an event log L, an infrequent trace σ, a frequency threshold of activity minActfreq, m frequent subpatterns $\text{Patt}_j\,(1 \le j \le m)$, conditional dependency probability threshold θ_{dep}.
Output: a Boolean value indicating whether the trace σ is infrequent but correct or not.
Step 1: *for* $i = 1$ *to* $|\sigma|$ *do*
Step 2: *compute* Actfreq$(\sigma[i])$ using Definition 9//determine whether there exist infrequent activities in the trace or not
Step 3: *if* σ_{l-1} < minActfreq *then*
//If the infrequent activity exists, it is necessary to determine whether there exists a special conditional dependency between the activity and the previous subsequence
Step 4: let $\widetilde{\sigma} = \sigma_1'$, $w = \sigma[i]$ (where $\sigma = \cdots \sigma_1' \bullet \sigma[i] \cdots$)
Step 5: *if* (a special data dependency C exists between $\widetilde{\sigma}$ and w in the trace σ) *then*
Step 6: compute CDP$(\widetilde{\sigma} \Rightarrow w)$ using Definition 12
Step 7: *if* CDP$(\widetilde{\sigma} \Rightarrow w) \ge \theta_{\text{dep}}$ *then*
Step 8: *return* true;
Step 9: *else return* false;
Step 10: *else return* false;
Step 11: *else* i++//continue to determine whether the next activity is an infrequent activity
//If an infrequent activity does not exist, check whether an infrequent subsequence in the trace exists
Step 12: according to the activities in frequent subpatterns, the trace σ can be divided into several subsequences, let $\sigma = \sigma_1 \bullet \sigma_2 \bullet \cdots \bullet \sigma_n$
Step 13: *for* (each σ_i) *do*
Step 14: *if* $\exists \text{Patt}_j\,(1 \le j \le m)$ such that $\sigma_i \sqsubseteq \text{Patt}_j$ *then*
Step 15: i++;
//Case 1: an infrequent subsequence exists, i.e., a subsequence does not match any of the frequent subpatterns
Step 16: *else* (*for* $\forall \text{Patt}_j\,(1 \le j \le m)$, $\sigma_i \not\sqsubseteq \text{Patt}_j$)
let $\sigma\prime \subset \sigma_i$ and $\sigma\prime \bullet a \subset \sigma_i$(or $a \subset \sigma_i[1]$), $\sigma\prime \bullet a$(or $\sigma_{i-1} \bullet a$) is an infrequent subsequence,
let $\widetilde{\sigma} = \sigma\prime$(or σ_{i-1}), $w = a$,
Step 17: *if* (a special data dependency C exists between $\widetilde{\sigma}$ and w in a trace σ) *then*
Step 18: computing CDP$(\widetilde{\sigma} \Rightarrow w)$ using Definition 12
Step 19: *if* CDP$(\widetilde{\sigma} \Rightarrow w) \ge \theta_{\text{dep}}$ *then* return to Step 15
Step 20: *else* return false;
Step 21: *else* return false;
Step 22: *if* $(i == n)$
//Check whether the interaction behavior between all legal subsequences is consistent with the interaction behavior profile between frequent subpatterns
Step 23: *for* σ
Step 24: *if* (the interaction behavior between σ_k and σ_{k+1} is not consistent with the interaction behavior profile between corresponding frequent subpatterns)
Step 25: *then* let $\widetilde{\sigma} = \sigma_k$, $w = \sigma_{k+1}[1]$ (where $\sigma_{k+1}[1]$ represents the first activity of σ_{k+1})
Step 26: *if* (a special data dependency C exists between $\widetilde{\sigma}$ and w in trace σ) *then*
Step 27: compute CDP$(\widetilde{\sigma} \Rightarrow w)$ using Definition 12
Step 28: *if* CDP$(\widetilde{\sigma} \Rightarrow w) \ge \theta_{\text{dep}}$ *then*
Step 29: k++
Step 30: *else return* false
Step 31: *else return* false
Step 32: *else* k++;//if consistent, continue to judge the behavior relationship between the next adjacent subsequence
Step 33: *if* $(k == n - 1)$ *then return* true;//the interaction behavior between all subsequences is reasonable

Algorithm 1: An effective infrequent behavior recognition approach based on frequent patterns under data awareness.

The results show that when the condition dependence threshold $\theta_{\text{dep}} = 0.7$, the traces σ_8, σ_9, and σ_{12} are considered to be an effective infrequent behavior using the proposed approach.

Subsequently, we evaluate the ability to identify effective infrequent behaviors in the proposed approach compared to the *IMi* algorithm [17], the *FM* algorithm [19], and the DHM algorithm [20]. Table 9 indicates that the proposed approach can correctly identify more effective infrequent behaviors than other approaches, whereas the DHM algorithm may mistake the incorrect infrequent trace as the correct one.

Finally, the optimization process model M_2 of the fusion control flow and data flow is constructed by incorporating these infrequent behaviors into the process modelM_1, as shown in Figure 10. The transitions $d_i\,(1 \le i \le 6)$ in process model M_2 are unobservable activities representing data flow that have been added for routing purposes only and do not appear in the event log. In adopting the approach proposed in [41], the fitness of the model M_2 is improved to 0.993, while the fitness of the initial model M_1 is 0.939.

Algorithm 1 uses two thresholds, a frequency threshold of activity and conditional dependency probability threshold. The former is used to differentiate the frequent activities and infrequent activities, while the latter is used to differentiate effective infrequent behaviors and noneffective infrequent behaviors. Actually, the performance of identifying effective

Input: an event log L, a frequency threshold of a trace minfreq
Output: a process model integrating data flow and control flow
Step 1: *for* (each trace σ in L)*do*
Step 2: *if* $|\sigma| \geq$ minfreq *then*
Step 3: FilterLog = FilterLog $\cup \sigma$
Step 4: *else* Outlier = Outlier $\cup \sigma$
//calculate the start activities and the end activities of frequent traces
Step 5: the initial process model M_1 is constructed by applying the *IMi* algorithm on the set FilterLog
Step 6: *for* (each trace σ in FilterLog) *do*
Step 7: compute first Act (σ_i), lastAct (σ_i)
Step 8: compute $\text{startActs}_{\text{FilterLog}} = \cup_{\forall \sigma_i \in L} \text{firstAct}(\sigma_i)$ and $\text{endActs}_{\text{FilterLog}} = \cup_{\forall \sigma_i \in L} \text{last Act}(\sigma_i)$
Step 9: *for* (each trace σ in Outlier) *do*
Step 10: *if* ($\sigma[1] \notin \text{startActs}_{\text{FliterLog}}$ or $\sigma[n] \notin \text{lastActs}_{\text{FliterLog}}$), where $|\sigma| = n$ *then*
Step 11: Noise = Noise $\cup \sigma$ and Outlier = Outlier $- \sigma$
Step 12: *for* (each trace σ in Outlier) *do*
Step 13: *if* IsInfrequent (σ) == true *then*
Step 14: Infrequent = Infrequent $\cup \sigma$ and Outlier = Outlier $- \sigma$
else Noise = Noise $\cup \sigma$ and Outlier = Outlier $- \sigma$
Step 15: an optimization process model M_2 is obtained by adding data flow and control flow to the initial process model M_1 to incorporate all infrequent but correct behaviors in Infrequent

Algorithm 2: An optimization approach for mining of process model with infrequent behaviors integrating data flow and control flow.

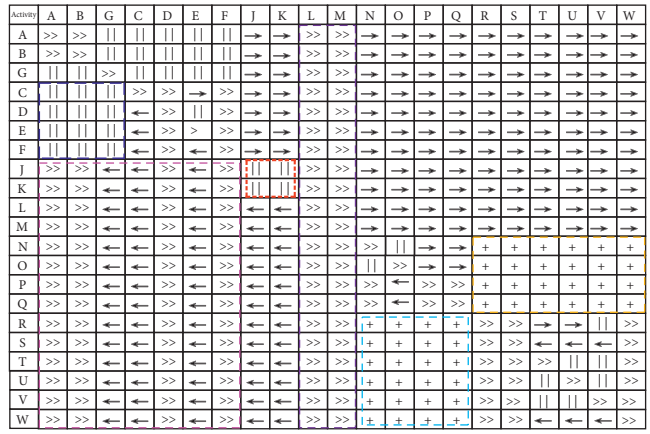

Figure 7: Behavior profile of frequent traces in the log of scheduled tickets in the train booking system.

infrequent behaviors is mainly affected by the conditional dependency probability threshold. To illustrate how varying this threshold affects the identification of effective infrequent behaviors, we designed the experiment to measure the amount of effective infrequent behavior correctly identified by our technique. Here, we use the previous synthetic log to evaluate the effect of different levels of threshold on Algorithm 1 by incrementally injecting infrequent behaviors. As shown in Figure 11, the results show that generally the rate of effective infrequent behavior correctly recognized decreases as the threshold parameter increases. When the threshold is set at a high value, these infrequent behaviors which rarely happen (i.e., only one or two times), and some infrequent behaviors which include other traces of recorded errors with the same data dependency conditions cannot be identified correctly.

6.2. Real-Life Dataset. We designed a simulation experiment for analysis using the claims data package provided by an insurance company platform. The data are from the company's Insurance Service Platform-Log Data1, including 980 cases, 13,280 events, 27 activities, and 12 attributes. For Log Data1, we compare the proposed approach, the IMi algorithm, and the DHM algorithm on precision [42] and fitness [41] value to evaluate the quality of the discovered model by injecting 1% to 9% infrequent behavior into the event log. In many cases, there is a trade-off between these two metrics. To balance them, the F-score is often used to combine fitness and precision through their harmonic means $2 \times (\text{fitness} * \text{precision}/\text{fitness} + \text{precision})$. The abscissa corresponds to the ratio of the injected infrequent behavior, and the ordinate corresponds to the fitness, precision, and F-score value, in Figures 12–14, respectively.

Figure 12 shows that the proposed approach can find more infrequent behaviors than the IM*i* and DHM approaches, and it significantly improves the fitness of the discovered model. Since the *IMi* algorithm only filters infrequent behaviors based on frequency from the control-

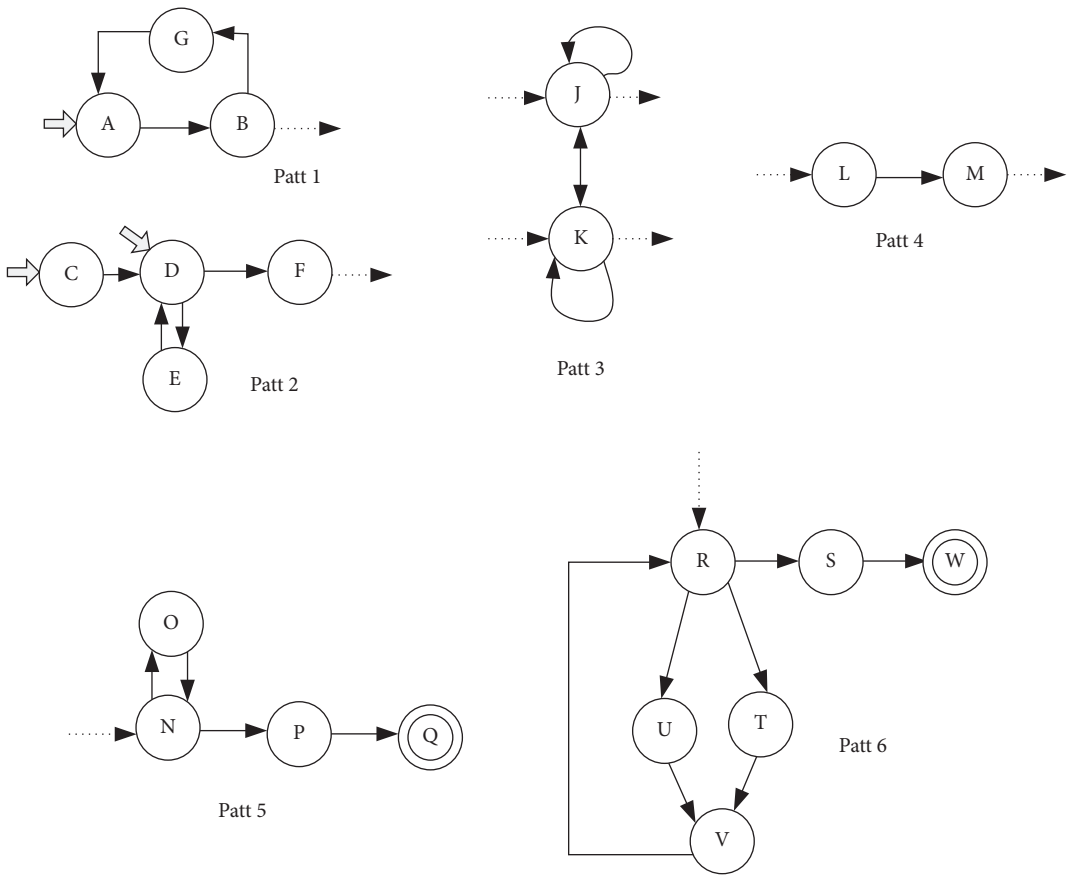

Figure 8: Maximal frequent patterns.

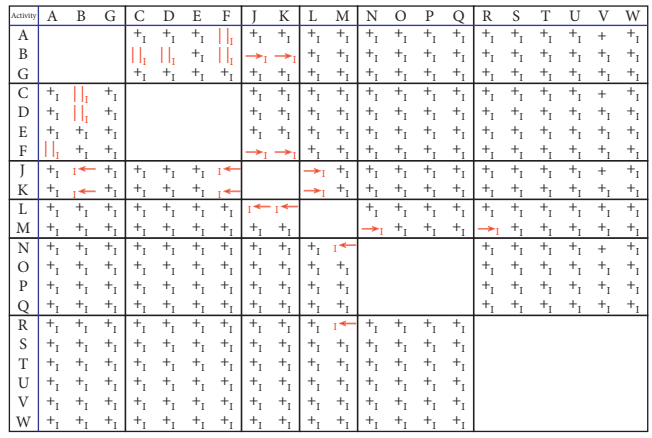

Figure 9: Interactive behavior profile of frequent patterns.

Table 3: Attributes of some activities and their value in the trace σ_8.

Identifier	Activity	State	Related persons	Wait for payment	Start time	Finish time
D	Login					
F	Success	Login success	=15	no		
J	Select a passenger					
K	Insert a passenger	Failure				
K'	Remove a passenger	Success				
K	Insert a passenger	Success				
L	Submit a reservation					
M	Confirm a reservation					
R	Online payment				2017-12-27 9:20	
S	Success to payment					2017-12-27 9:23
W	Reservation completion					

Table 4: Attributes of some activities and their value in traceσ_9.

Identifier	Activity	State	Related persons	Wait for payment	Start time	Finish time
D	Login					
F	success	Login success	<15	no		
J	Select a passenger					
L	Submit a reservation					
M	Confirm a reservation					
R	Online payment				2017-12-25 13:20	
T	Wait for payment					2017-12-25 13:50
Q	Success to cancel					

Table 5: Attributes of some activities and their value in trace σ_{10}.

Identifier	Activity	state	Related persons	Wait for payment	Start time	Finish time
D	Login					
F	Success	Login success	<15	No		
J	Select a passenger					
L	Submit a reservation					
M	Confirm a reservation					
R	Online payment				2017-12-25 19:03	
T	Wait for payment					2017-12-25 19:33
W	Reservation completion					

Table 6: Attributes of some activities and their value in trace σ_{11}.

Identifier	Activity	State	Related persons	Wait for payment	Start time	Finish time
D	Login					
F	Success	Login success	<15	No		
J	Select a passenger					
L	Submit a reservation					
M	Confirm a reservation					
N	Cancel an reservation					
R	Online payment				2017-12-29 10:26	
S	Success to payment					2017-12-29 10:30
W	Reservation completion					

Table 7: Attributes of some activities and their value in trace σ_{12}.

Identifier	Activity	State	Related persons	Wait for payment	Start time	Finish time
A	Search					
B	Booking					
D	Login					
E	Failure	Login failure				
F	Success	Login success	<15	Yes		
R	Online payment					
S	Success to payment				2017-12-30 14:13	
W	Reservation completion					

Table 8: The conditional dependence probabilities are obtained according to Algorithm 1.

Trace	Compliant pattern	Infrequent activity/the pair	$\tilde{\sigma}$	w	Particular data dependence C	CDP$(\tilde{\sigma} \Rightarrow tw)$
σ_8	Patt 1, Patt 2, Patt 3	$K\prime$	ABDFJK	$K\prime$	F (related persons) = 15	0.941
σ_9	Patt 1, Patt 2, Patt 3, Patt 4	TQ	ABDFJLMRT	Q	T (finish time) − R (start time) = 30 minutes	0.863
σ_{10}	Patt 2, Patt 1, Patt 3, Patt 4	TW	DFABJLMRT	W	T (finish time) − R (start time) = 30 minutes	−0.863
σ_{11}	Patt 1, Patt 2, Patt 3	ML			No	0
σ_{12}	Patt 1, Patt 2, Patt 6	FR	ABDEDF	R	F (wait for payment) = yes	0.944

Table 9: The results of effective infrequent behaviors obtained by several algorithms.

Trace	Process mining algorithm (threshold value = 0.7)			
	The proposed approach	IMi	MF	DHM
σ_8	Useful	Useless	Useless	Useless
σ_9	Useful	Useless	Useless	Useless
σ_{10}	Useless	Useless	Useless	Useful
σ_{11}	Useless	Useless	Useless	Useless
σ_{12}	Useful	Useful	Useless	Useful

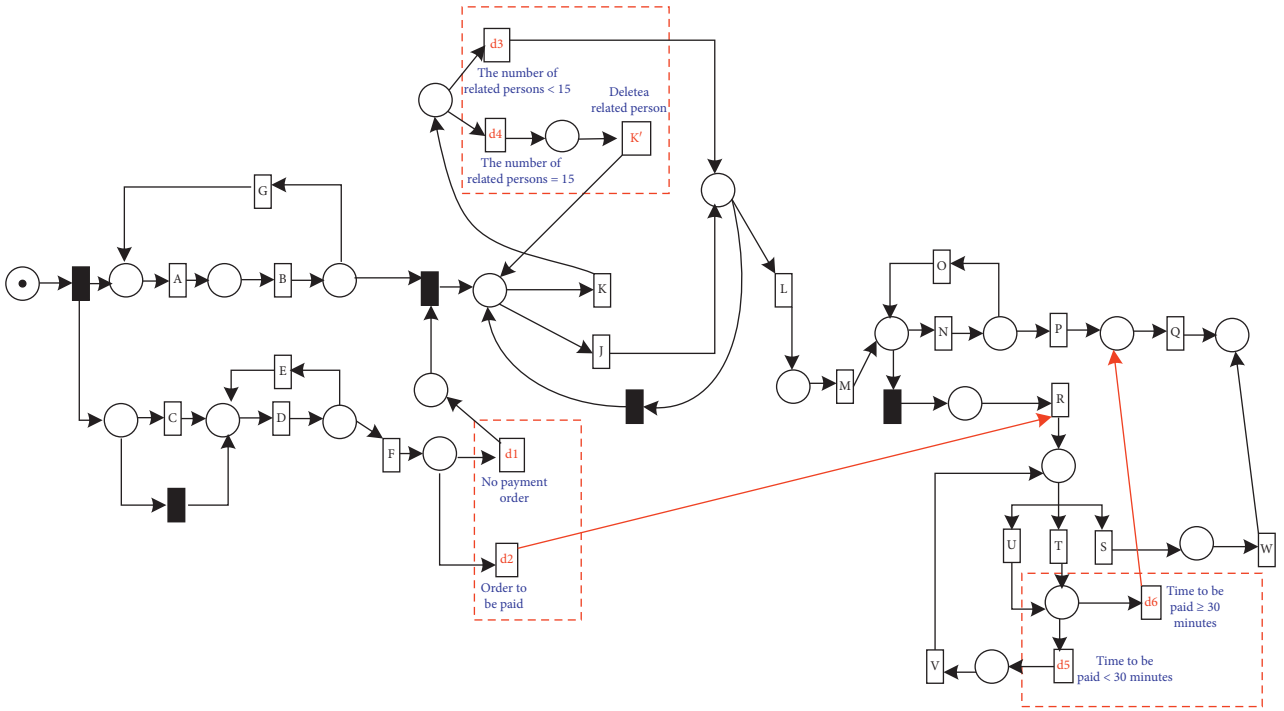

Figure 10: An optimization process model M_2 integrating control flow and data flow.

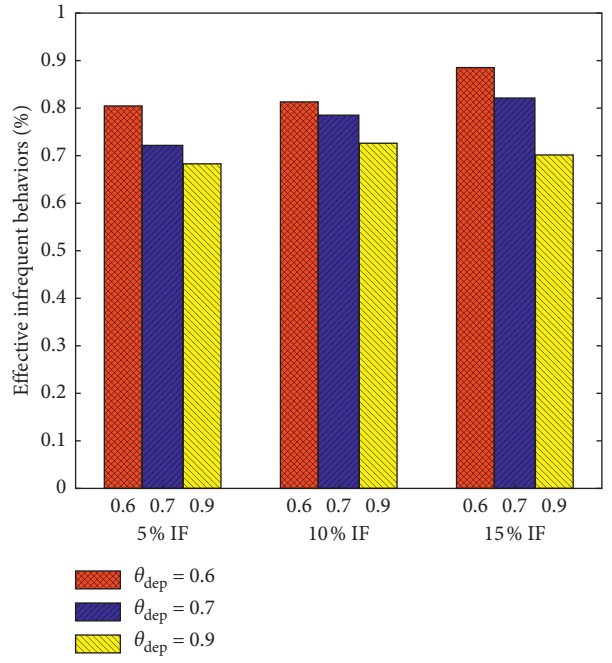

Figure 11: Comparison on the rate of effective infrequent behavior correctly recognized using different thresholds.

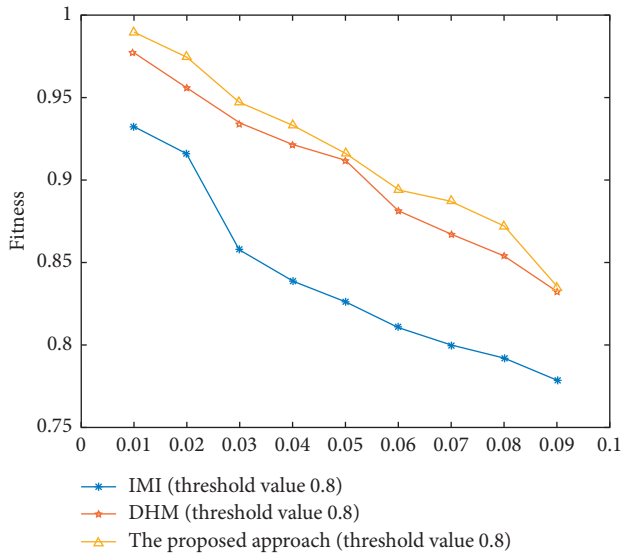

Figure 12: Fitness comparison on real-life logs.

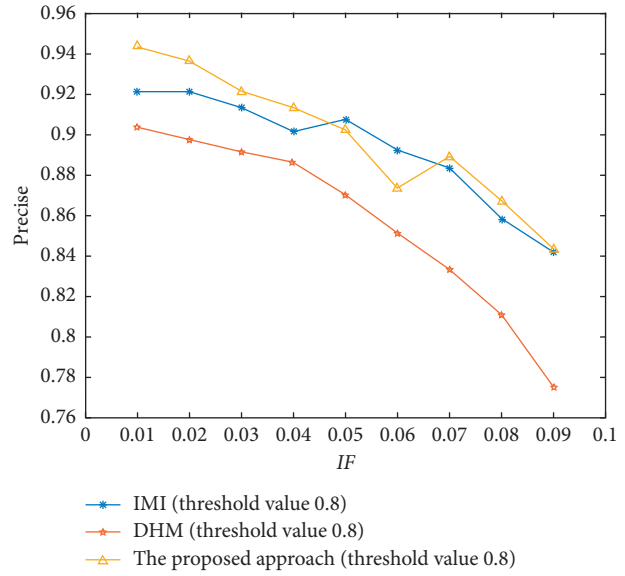

Figure 13: Precision comparison on real-life logs.

flow perspective, many infrequent behaviors are disregarded as noise. Therefore, the fitness of the resulting model is relatively low. With the increase in infrequent behavior, the overall fitness of the three approaches declines.

Figure 13 indicates that the precision of the model obtained by the proposed approach is higher than that of the others when injecting less infrequent behaviors. This may be due to a reduction in additional behaviors by adding data flow to the resulting model. The precision of the DHM algorithm is relatively low. Although it can find more infrequent behavior, more control flow is added in the resulting causal net without data flow information, which makes the discovered model more complicated. With the increase in infrequent behavior, the overall precision of the three approaches shows a downward trend.

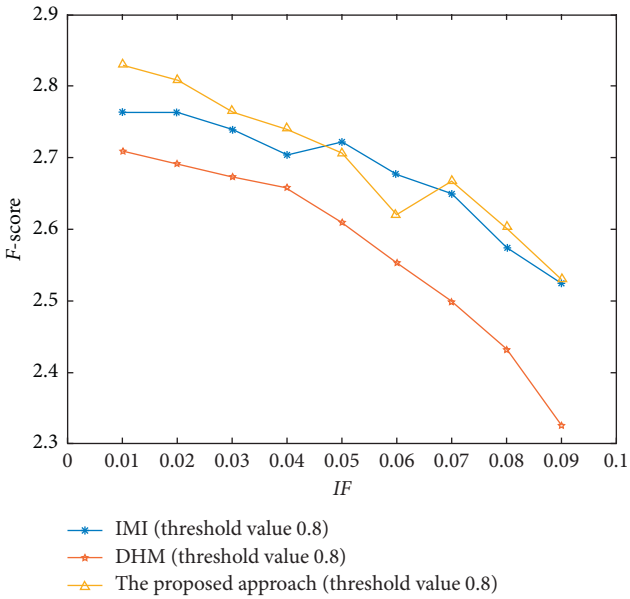

Figure 14: *F*-score comparison on real-life logs.

Figure 14 shows that the *F*-score for the proposed approach is generally superior to the IMI and DHM approaches. As the increase in infrequent behavior may lead to a small decrease in precision, in some cases, it will be slightly lower than others.

The experimental results of synthetic and real logs show that our approach has a noticeable improvement over the fitness of the discovered process model without significantly reducing the precision. Thus, our approach is promising that can preserve the effective infrequent behavior representing important information of the system when discovering the process model. Hence, the proposed approach provides better support for enterprise business improvement.

7. Conclusions and Future Work

In this paper, an effective infrequent behavior recognition approach based on frequent patterns under data awareness is presented. It analyzes the coupling between infrequent behavior and the data dependency information and uses the conditional dependence probability to quantify the influence strength between them. This approach provides a qualitative and quantitative analysis for the identification of effective infrequent behavior and realizes long-term dependencies between the activity and the frequent pattern, not only directly-follows data dependencies. Moreover, an optimization approach for mining of process models with infrequent behaviors integrating data flow and control flow is provided in this paper. We compared the proposed approach with other techniques, showing that our approach discovers infrequent behavior that other techniques cannot detect. Furthermore, the evaluation on synthetic and real-life event logs indicates that incorporating infrequent but correct behavior will greatly improve the fitness of the discovered process model without significantly reducing its precision by adding appropriate data flow and control flow to the resulting process model.

In the future, the proposed approach will be applied to more application fields, and various factors that lead to infrequent behavior occurrence from a data-flow perspective will be further studied. Association rules will be used to reveal data dependency between activities to provide a better basis for the recognition of infrequent behaviors.

Data Availability

The data used to support the findings of this study were supplied by an insurance company under license and so cannot be made freely available. Requests for access to these data should be made to slxjx@aust.edu.cn.

Conflicts of Interest

The authors declare that they have no potential conflicts of interest.

Acknowledgments

This work was partially supported by the National Natural Science Foundation, China (nos. 61572035 and 61402011), the Leading Backbone Talent Project in Anhui Province, China (2020-1-12), the Natural Science Foundation of Anhui Province, China (no. 2008085QD178), Anhui Province Academic and Technical Leader Foundation (no. 2019H239), Anhui Province College Excellent Young Talents Fund Project of China (gxyqZD2020020), and the Open Project of the Key Laboratory of Embedded System and Service Computing Ministry of Education (no. ESSCKF2018-04).

References

[1] W. M. P. Van der Aalst, *Process Mining-Data Science in Action*, Springer, Berlin, Germany, 2nd edition, 2016.

[2] W. Van Der Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business processes*, Springer, Heidelberg, Germany, 2011.

[3] J. C. A. M. Buijs, B. F. Van Dongen, and W. M. P. van Der Aalst, "On the role of fitness, precision, generalization and simplicity in process discovery," in *Proceedings of the On the Move to Meaningful Internet Systems: OTM 2012*, pp. 305–322, Springer, Rome, Italy, September 2012.

[4] L. Măruşter, A. T. O. N. Weijters, W. M. P. Van Der Aalst et al., "A rule-based approach for process discovery: dealing with noise and imbalance in process logs," *Data Mining and Knowledge Discovery*, vol. 13, no. 1, pp. 67–87, 2006.

[5] S. Suriadi, R. Andrews, A. H. M. ter Hofstede, and M. T. Wynn, "Event log imperfection patterns for process mining: towards a systematic approach to cleaning event logs," *Information Systems*, vol. 64, pp. 132–150, 2017.

[6] W.-S. Yang and S.-Y. Hwang, "A process-mining framework for the detection of healthcare fraud and abuse," *Expert Systems with Applications*, vol. 31, no. 1, pp. 56–68, 2006.

[7] D. Lo, H. Cheng, J. Han et al., "Classification of software behaviors for failure detection: a discriminative pattern mining approach," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 557–566, ACM, Paris France, June 2009.

[8] W. Van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: discovering process models from event logs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 9, pp. 1128–1142, 2004.

[9] A. K. Alves de Medeiros, B. F. van Dongen, W. M. P. van der Aalst, and A. J. M. M. Weijters, "Process mining: extending the alpha-algorithm to mine short loops," *BETA Working Paper Series 113*, TU Eindho-ven, Eindhoven, Netherlands, 2004.

[10] L. Wen, W. M. P. van der Aalst, J. Wang, and J. Sun, "Mining process models with non-free-choice constructs," *Data Mining and Knowledge Discovery*, vol. 15, no. 2, pp. 145–180, 2007.

[11] J. M. E. M. van derWerf, B. F. van Dongen, C. A. J. Hurkens et al., "Process discovery using integer linear programming," *FundamentaInformaticae*, vol. 94, no. 3-4, pp. 387–412, 2009.

[12] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Discovering block-structured process models from event logs-a constructive approach," in *Proceedings of the International Conference on Applications and Theory of Petri Nets and Concurrency*, pp. 311–329, Springer, Milan, Italy, June 2013.

[13] J. Carmona, J. Cortadella, and M. Kishinevsky, "A region-based algorithm for discovering petri nets from event logs," in *Proceedings of the International Conference on Business Process Management, Lecture Notes in Computer Science*, pp. 358–373, Springer, Milan, Italy, September 2008.

[14] A. Weijters, W. M. P. van Der Aalst, and A. K. A. De Medeiros, "Process mining with the heuristics miner-algorithm," Tech. Rep. WP, Technische Universiteit Eindhoven, Eindhoven, Netherlands, 2006.

[15] A. Weijters and J. T. S. Ribeiro, "Flexible heuristics miner (FHM)," in *Proceedings of the 2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pp. 310–317, IEEE, Paris, France, April 2011.

[16] S. K. L. M. vanden Broucke and J. De Weerdt, "Fodina: a robust and flexible heuristic process discovery technique," *Decision Support Systems*, vol. 100, pp. 109–118, 2017.

[17] C. W. Günther and W. M. P. Van Der Aalst, "Fuzzy mining-adaptive process simplification based on multi-perspective metrics," in *Proceedings of the International Conference on Business Process Management, Lecture Notes in Computer Science*, pp. 328–343, Springer, Brisbane, Australia, September 2007.

[18] V. Liesaputra, S. Yongchareon, and S. Chaisiri, "Efficient process model discovery using maximal pattern mining," in *Proceedings of the International Conference on Business Process Management*, pp. 441–456, Springer, Rio de Janeiro, Brazil, September 2016.

[19] S. Goedertier, D. Martens, J. Vanthienen et al., "Robust process discovery with artificial negative events," *Journal of Machine Learning Research*, vol. 10, pp. 1305–1340, 2009.

[20] H. Ponce-de-León, J. Carmona, and S. K. L. M. vanden Broucke, "Incorporating negative information in process discovery," in *Proceedings of the International Conference on Business Process Management*, pp. 126–143, Springer, Vienna, Austria, September 2016.

[21] J. C. A. M. Buijs, B. F. van Dongen, and W. M. P. van der Aalst, "A genetic algorithm for discovering process trees," in *Proceedings of the 2012 IEEE Congress on Evolutionary Computation*, pp. 1–8, IEEE, Brisbane, Australia, June 2012.

[22] A. J. Rembert, A. Omokpo, P. Mazzoleni, and R. T. Goodwin, "Process discovery using prior knowledge," in *Proceedings of*

the International Conference on Service-Oriented Computing, pp. 328–342, Springer, Dubai, UAE, December 2013.

[23] E. Bellodi, F. Riguzzi, and E. Lamma, "Statistical relational learning for workflow mining," *Intelligent Data Analysis*, vol. 20, no. 3, pp. 515–541, 2016.

[24] D. Chapela-Campa, M. Mucientes, and M. Lama, "Discovering infrequent behavioral patterns in process models," in *Proceedings of the International Conference on Business Process Management*, pp. 324–340, Springer, Barcelona, Spain, September 2017.

[25] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Discovering block-structured process models from event logs containing infrequent behavior," in *Proceedings of the International Conference on Business Process Management*, pp. 66–78, Springer, Beijing, China, August 2013.

[26] R. Conforti, M. La Rosa, and A. H. M. ter Hofstede, "Filtering out infrequent behavior from business process event logs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 2, pp. 300–314, 2016.

[27] M. F. Sani, S. J. van Zelst, and W. M. P. van der Aalst, "Improving process discovery results by filtering outliers using conditional behavioural probabilities," in *Proceedings of the International Conference on Business Process Management*, pp. 216–229, Springer, Barcelona, Spain, September 2017.

[28] H. Sun, W. Liu, L. Qi, Y. Du, X. Ren, and X. Liu, "A process mining algorithm to mixed multiple-concurrency short-loop structures," *Information Sciences*, vol. 542, pp. 453–475, 2021.

[29] H. Sun, Y. Du, L. Qi, and Z. He, "A method for mining process models with indirect dependencies via petri nets," *IEEE Access*, vol. 7, pp. 81211–81226, 2019.

[30] B. F. Van Dongen, A. K. Alves de Medeiros, and L. Wen, "Process mining: overview and outlook of petri net discovery algorithms," *Transactions on Petri Nets and Other Models of Concurrency II*, Springer, Berlin, Germany, pp. 225–242, 2009.

[31] L. Ghionna, G. Greco, A. Guzzo et al., "Outlier detection techniques for process mining applications," in *Proceedings of the International Symposium on Approachologies for Intelligent Systems*, pp. 150–159, Springer, Toronto, Canada, May 2008.

[32] F. Bezerra and J. Wainer, "Algorithms for anomaly detection of traces in logs of process aware information systems," *Information Systems*, vol. 38, no. 1, pp. 33–44, 2013.

[33] X. Lu, D. Fahland, F. J. H. M. van den Biggelaar, and W. M. P. van der Aalst, "Detecting deviating behaviors without models," in *Proceedings of the International Conference on Business Process Management*, pp. 126–139, Springer, Toronto, Canada, May 2016.

[34] F. Mannhardt, M. de Leoni, H. A. Reijers, and W. M. P. van der Aalst, "Data-driven process discovery-revealing conditional infrequent behavior from event logs," in *Proceedings of the International Conference on Advanced Information Systems Engineering*, pp. 545–560, Springer, Essen, Germany, June 2017.

[35] S. Schönig, C. Di Ciccio, F. M. Maggi, and J. Mendling, "Discovery of multi-perspective declarative process models," in *Proceedings of the International Conference on Service-Oriented Computing*, pp. 87–103, Springer, Dubai, UAE, December 2016.

[36] M. L. Bernardi, M. Cimitile, C. Di Francescomarino, and F. M. Maggi, "Using discriminative rule mining to discover declarative process models with non-atomic activities," in *Proceedings of the International Symposium on Rules and Rule Markup Languages for the Semantic Web*, pp. 281–295, Springer, Galway, Ireland, November 2014.

[37] V. Leno, M. Dumas, F. M. Maggi, M. La Rosa, and A. Polyvyanyy, "Automated discovery of declarative process models with correlated data conditions," *Information Systems*, vol. 89, p. 101482, 2020.

[38] W. M. P. van der Aalst, M. Dumas, F. Gottschalk et al., "Preserving correctness during business process model configuration," *Formal Aspects of Computing*, vol. 22, no. 3-4, pp. 459–482, 2010.

[39] S. Smirnov, M. Weidlich, and J. Mendling, "Business process model abstraction based on behavioral profiles," in *Proceedings of the International Conference on Service-Oriented Computing*, pp. 1–16, Springer, San Francisco, CA, USA, December 2010.

[40] M. Weidlich, A. Polyvyanyy, N. Desai, and J. Mendling, "Process compliance measurement based on behavioural profiles," in *Proceedings of the International Conference on Advanced Information Systems Engineering*, pp. 499–514, Springer, Hammamet, Tunisia, June 2010.

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

[42] A. Adriansyah, J. Munoz-Gama, J. Carmona et al., "Alignment based precision checking," in *Proceedings of the International Conference on Business Process Management*, pp. 137–149, Springer, Tallinn, Estonia, September 2012.