Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 2545413, 18 pages
https://doi.org/10.1155/2020/2545413

Research Article

Analysis and Application of Transition Systems Based on Petri Nets and Relation Matrices to Business Process Management

Dong Han [1] **and Yinhua Tian** [2]

[1]*College of Energy and Mining Engineering, Shandong University of Science and Technology, Qingdao 266590, China*
[2]*Department of Information Engineering, Shandong University of Science and Technology, Taian 271000, China*

Correspondence should be addressed to Yinhua Tian; skdxxtyh@163.com

Received 21 December 2019; Accepted 13 February 2020; Published 31 March 2020

Academic Editor: Jean Jacques Loiseau

In order to improve the efficiency of conformance checking in business process management, a business alignment approach is presented based on transition systems between relation matrices and Petri nets. Firstly, a log-based relation matrix of the events is obtained according to the event log. Then, the events in the relation matrix are observed and the transitions in the model are firing, and the activities in the log and in the model are compared. Next, the states of the log and the model are recorded until no new state can be generated, so a transition system can be obtained which includes optimal alignments between the event log and the process model. Finally, two detailed algorithms are presented to obtain an optimal alignment and all optimal alignments between the trace and the model based on the given cost function, respectively. The availability and effectiveness of the proposed approach are proved theoretically.

1. Introduction

Business process management (BPM) aims to provide a unified modeling, running, and monitoring environment for business processes from information technology and management technology. As an important branch of BPM, process mining is to discover, monitor, and enhance the actual business processes by extracting valuable information from event logs [1]. The research on process mining is significant for the implementation, analysis, and improvement of business processes, so it is a hot topic in the related fields [2, 3]. Process mining mainly includes process discovery, conformance checking, and process enhancement [4–6]. Conformance checking is to compare the events in the event logs with the activities in the process models, and it can find the similarities and differences between the observed behaviors and the modeled behaviors [7–10].

Among many conformance checking approaches, aligning observed and modeled behaviors becomes an important means to measure the compliance of event logs and process models [11–13]. The alignment approaches aim at finding the deviations between process models and event logs. Usually, the alignments with the least deviation are considered as the optimal alignments. The search algorithms of all the optimal alignments are NP-hard problems, whose time complexity and space complexity are very high. In the research field of process mining, alignment approaches have been deeply studied and widely applied [14–19]. There are a large amount of literatures on alignment approaches to introduce their ideas, motivations, and problems resolved.

Through the analysis and conclusion of various alignment approaches [20–24], we find the existing problems of the current ones, mainly including high complexity of the search algorithms, unable to find the required and accurate optimal alignments, and unable to find all the optimal alignments. To solve the abovementioned problems, we propose an alignment approach based on relation matrices and Petri nets. The proposed approach is completely different from the existing alignment approaches in which it does not deal with one trace but all the traces in the event log at once. In this approach, the relation matrix reflects all the

partial order relations between events in the event log. A transition system can be obtained by comparing the events in the relation matrix with the activities in the Petri net, which includes all the alignments between all the traces and the process model.

Our approach includes two steps: one is to generate the search space; the other is to search the optimal alignments in the space. Compared with other alignment approaches, this approach has two advantages: one is that it can calculate the accurate alignment results but not approximate solutions; the other is that it can obtain all the optimal alignments based on the given cost functions. However, the approach presented in this paper can embody the alignment results between all traces in the event log and the process model in a transition system; thus, it can save time and space to calculate the search space.

The rest of this paper is organized as follows. Section 2 discusses the related work. Section 3 recalls some basic concepts. The generation algorithm of the alignment transition system is presented in Section 4. Section 5 presents the approaches of searching for an optimal alignment and all optimal alignments in the alignment transition system, respectively. Section 6 describes the scalability of our approach. Case studies are given to illustrate the superiority of our approach in Section 7. Section 8 draws the conclusion and the future work.

2. Related Work

The computation of the optimal alignments is NP-hard, which has very high time and space complexity. Now, the existing alignment approach can only deal with one trace once. If we want to compute all the optimal alignments for the whole event log, we must do the same work for several times. As related work, the following approaches are introduced in this paper.

An approach is presented by Cook and Wolf, which compares the traces with process models in order to quantitatively measure their similarity [19]. However, the approach is analyzed using the state-space technology, and it does partially support the invisible transitions and duplicate transitions. In addition, a heuristics estimation function is used to deal with the high computational complexity in this approach. The function simplifies the search space. However, the application of the function may lead to the approximate optimal solutions.

An approach to align observed and modeled behaviors is proposed by Adriansyah [13]. The approach is a very classical one in the related work because it can obtain all the exact optimal alignments between the given trace and the process model. Its main idea is as follows: ① an event net is constructed based on the trace; ② the product of the event net and the process net is generated, which is also a Petri net; ③ part of the reachable graph of the product model is constructed while searching for the shortest path using A^* algorithm, which is also conformed to the definition of the transition system; ④ an optimal alignment can be obtained when arriving at the final state. The time and space complexity is very high in order to obtain the solutions.

An approach of conformance checking based on partially ordered event data is proposed by Lu et al. [20, 21]. The main idea of the approach is as follows: ① the partially ordered traces are extracted from the existing logs; ② the partially ordered alignments are obtained through dealing with the partially ordered traces; ③ a quantitative-based quality metric is introduced to objectively compare the eventual results of conformance checking. Although the alignment procedure is simplified to some extent using the technology of partially ordered alignments, only the approximate solution of the optimal alignment can be obtained in some cases.

A workflow decomposition approach to align observed and modeled behaviors is proposed by Wang et al. [22]. The approach can divide the large process models and the relevant event logs into several separate parts that can be analyzed and aligned independently. However, the approach can only deal with the block workflow models which can be divided into several segments. Generally, it can only obtain some alignments, rather than all the optimal alignments.

An efficient alignment approach between event logs and process models is presented by Song et al. [23]. The approach leverages effective heuristics and trace replaying to significantly reduce the overall search space for seeking the optimal alignment. The approach improves the efficiency of computing the search space, including the time aspect and space aspect. However, there are still some redundant nodes in the search space, which are not on the paths to the optimal alignments. In addition, only part of the optimal alignments can be obtained due to the limitation of the preprocess, even in some cases only approximate optimal alignments can be obtained.

A reduced alignment approach between event logs and process models is presented by Tian et al., which is named as OAT approach [24]. An optimal alignment tree is generated through this approach. In the optimal alignment tree, the optimal alignments can be easily found by adding the final marks to the leaf nodes which is related to the optimal alignments. The approach largely reduces the time complexity when searching for the optimal alignments in the search space. However, there are some fatal problems, for instance, all the information is placed on the nodes, the duplicate nodes are not shared, and the invalid nodes are not pruned. Hence, the size of the optimal alignment tree is too large and even the search space will explode.

The approach proposed in this paper can obtain the log-based relation matrix according to the given event log. The relation matrix can illustrate all the precursor and successor relations between the events in the log. Then, an alignment transition system can be acquired through comparing the events in the log with the activities in the model and predicting the next move. The alignment transition system includes all the optimal alignments between all traces and the model. Two algorithms are presented to calculate an optimal alignment and all the optimal alignments based on the cost function, respectively.

Compared with other approaches, the approach in this paper has two advantages. One is that it can obtain the accurate alignment results rather than the approximate

solutions. The other is that it can obtain all optimal alignments between traces and process models based on the given cost function. Especially, the search space of our approach includes the alignment results for all traces in the log, but not only one.

3. Preliminaries

This section briefly reviews some basic concepts, including traces [4], event logs [4], Petri nets [25–31], transition systems [13], and alignments [13].

Definition 1 (trace and event log). Let A be a set of activities. A trace $\sigma \in A^*$ is a sequence of activities. An event log $L \in \mathbb{B}(A^*)$ is a multiple set of traces on A.

Definition 2 (labeled Petri net system). Let A be a set of activities. A labeled Petri net system over A is a tuple $N = (P, T; F, \alpha, m_i, m_f)$, where

(1) P is the set of places

(2) T is the set of transitions, and $P \cup T \neq \emptyset$, $P \cap T = \emptyset$

(3) $F \subseteq (P \times T) \cup (T \times P)$ is an arc set between transitions and places, i.e., a flow relation

(4) α: $T \longrightarrow A^\tau$ is a function that maps transitions to labels, and τ denotes the invisible transition, $A^\tau = A \cup \{\tau\}$

(5) m_i and m_f are the initial marking and final marking, respectively

For convenience, in the remainder of this paper, the labeled Petri net system is abbreviated as Petri net.

Definition 3 (perset and postset). Let $N = (P, T; F, \alpha, m_i, m_f)$ be a Petri net. For $\forall x \in P \cup T$,

$$\begin{aligned} {}^{\cdot}x &= \{y \mid y \in P \cup T \wedge (y, x) \in F\}, \\ x^{\cdot} &= \{y \mid y \in P \cup T \wedge (x, y) \in F\}, \end{aligned} \tag{1}$$

where ${}^{\cdot}x$ represents the preset of x and $x^{\cdot}$ represents the postset of x.

We describe the transition firing rules by using the multisets of places. For any reachable state $m \in \mathbb{B}(P)$, the transition firing rules of Petri net $N = (P, T; F, \alpha, m_i, m_f)$ are as follows:

(1) For transition $t \in T$, if ${}^{\cdot}t \in m$, t is enabled denoted by $m[t>$

(2) If $m[t>$, it means that the transition t can occur under the marking m, and after the transition t is fired, a new marking m' is generated, denoted by $m[t>m'$, where $m' = m \uplus t^{\cdot} - {}^{\cdot}t$

The set of all reachable states from state m is denoted as $R(m)$, and $m \in R(m)$. In Petri net $N = (P, T; F, \alpha, m_i, m_f)$, m_i means the initial state of the system, and then $R(m_i)$ represents the set of all reachable states in the running process of the system.

Definition 4 (transition system). Let A be a set of activities. A transition system is a triplet $TS = (S, A, T)$, where S is the set of states, and $T \subseteq S \times A \times S$ is the set of transitions. $S^{\text{start}} \subseteq S$ is the set of initial states, and $S^{\text{end}} \subseteq S$ is the set of final states.

Definition 5 (alignment). Let A be a set of activities. $\sigma \in A^*$ is a trace over A and $N = (P, T; F, \alpha, m_i, m_f)$ is a Petri net over A. An alignment $\gamma \in (A^{\gg} \times T^{\gg})^*$ between σ and N is a legal movement sequence such that

(1) $\pi_1(\gamma)_{\downarrow A} = \sigma$, i.e., its sequence of movements in the trace (ignoring >>) yields the trace

(2) $m_i \longrightarrow^{\pi_2} (\gamma)_{\downarrow T} m_f$, i.e., its sequence of movements in the model (ignoring >>) yields a complete firing sequence of N

For all tuples $(a, t) \in \gamma$ in an alignment, (a, t) is one of the following movements:

(1) log move if $a \in A$ and $t = >>$

(2) model move if $a = >>$ and $t \in T$

(3) synchronous move if either $a \in A$ and $t \in T$ or $a = >>$ and $\alpha(t) = \tau$

(4) illegal move otherwise

We consider all of the log moves, model moves, and synchronous moves as legal ones. An alignment is legal if it only contains legal moves.

$\Gamma_{\sigma,N}$ is the set of all alignments between trace σ and model N.

There may be several different alignments between the trace and model. To get the most suitable alignments, a cost function $c((a, t))$ is used to assign a certain value to each move. According to the given cost function, the alignments with the least total cost are called optimal alignments.

Likelihood cost function $c()$ determines the optimal alignment set between the given trace and model directly. In this paper, the standard likelihood cost function $lc()$ is used to assign the cost to the moves, i.e., the cost value of the synchronous move, log move, and model move is 0, 1, and 1, respectively.

$\Gamma^o_{\sigma,N,lc}$ is the set of all optimal alignments between trace σ and model N based on the function $lc()$.

4. Generation of Transition Systems

When measuring the fitness between the event log and the process model, the main work is to align the events in the trace with the activities in the model. In the current alignment procedure, assuming that the observed event in the log is x, the fired transition in the model is t_i, and t_i's mapping activity is y. In the next procedure, one of the following three scenarios may occur: ① we can observe activity x in the log but x cannot be modeled by firing the transition in the model, then a log move $(x, >>)$ is generated; ② when activity y is modeled by firing the transition t_i in the model but cannot be observed in the log, a move $(>>, t_i)$ is generated; if y is equal to τ, $(>>, t_i)$ is a synchronous move; else, $(>>, t_i)$ is a model move; ③ if x is equal to y, a synchronous move (x, t_i) can be generated when the activity

observed in the log is the same as the one modeled in the model.

The proposed approach is derived from the above-mentioned idea. It aims to observe the event log, run the process model, and compare the event in the log with the activity in the model. We record the states of the log and model, and then we can get an optimal alignment graph. The graph includes all the optimal alignments.

Next, we take the given event log and process model as an example to introduce the basic principles of the proposed approach.

Let $A = \{a, b, c, d\}$ be a set of activities. There is an event log of a simple process on A, as shown in Table 1. We denote the event log as $L_1 = [\sigma_1, \sigma_2, \sigma_3, \sigma_4]$, where $\sigma_1 = <a, b>$, $\sigma_2 = <b, d>$, $\sigma_3 = <a, a, b>$, and $\sigma_4 = <a, b, d>$.

Given process model $N_1 = (P_1, T_1; F_1, \alpha_1, m_{i,1}, m_{f,1})$, as shown in Figure 1. Its place set is $P_1 = \{p_1, p_2, p_3, p_4\}$; its transition set is $T_1 = \{t_1, t_2, t_3, t_4\}$; its flow relation set is $F_1 = \{(p_1, t_1), (t_1, p_2), (p_2, t_2), (p_2, t_3), (t_2, p_3), (t_3, p_3), (p_3, t_4), (t_4, p_4)\}$; the mapping relations between transitions and activities are $\alpha_1(t_1) = a$, $\alpha_1(t_2) = b$, $\alpha_1(t_3) = c$, $\alpha_1(t_4) = d$; the initial marking is $m_{i,1} = [p_1]$; the final marking is $m_{f,1} = [p_4]$.

4.1. Log-Based Relation Matrices. Consider for instance $L_1 = [\sigma_1, \sigma_2, \sigma_3, \sigma_4]$ again. For this event log, the following log-based order relations can be found, as in (2)–(5):

$$>_{L_1} = \{(a, b), (b, d), (a, a)\}, \tag{2}$$

$$\longrightarrow_{L_1} = \{(a, b), (b, d)\}, \tag{3}$$

$$\#_{L_1} = \{(a, d), (b, a), (b, b), (d, a), (d, b), (d, d)\}, \tag{4}$$

$$\|_{L_1} = \{(a, a)\} \tag{5}$$

Relation $>_{L_1}$ contains all pairs of activities with a directly following relation. $a >_{L_1} b$ because b directly follows a in trace $\sigma_1 = <a, b>$. However, $b >_{L_1} a$ because a never directly follows b in any trace in the log. $\longrightarrow_{L_1}$ contains all pairs of activities in a causal relation, e.g., $a \longrightarrow_{L_1} b$ because sometimes b directly follows a and never the other way around ($a >_{L_1} b$ and $b >_{L_1} a$). $a \#_{L_1} d$ because $a >_{L_1} d$ and $d >_{L_1} a$. $a \|_{L_1} a$ because $a >_{L_1} a$, i.e., a follows itself.

We can conclude the footprint of the log L_1, as in (6). The subscripts have been removed in

$$\begin{array}{c} \\ a \\ b \\ d \end{array} \begin{array}{c} \begin{array}{ccc} a & b & d \end{array} \\ \begin{bmatrix} \| & \rightarrow & \# \\ \# & \# & \rightarrow \\ \# & \# & \# \end{bmatrix} \end{array}. \tag{6}$$

According to the footprint in (6), we can get all the relations of the activities in the event log, but not only a trace. Although the footprint embodies the relations between the activities in the log, it cannot describe the current states of the log, especially the start and end of the traces. To illustrate all the information needed in the alignment process, we convert the traces in the log into the transition systems, as shown in Figure 2.

Table 1: The event log of a simple process.

Case id	Event id
1	a
2	b
3	a
3	a
1	b
3	b
4	a
4	b
2	d
4	d

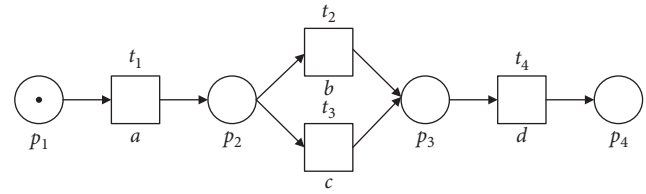

Figure 1: Process model N_1.

A trace in the log is corresponding with a transition system, which is called as trace-based transition system. Firstly, we create an initial state, denoted as s_0, which means the start of the trace. Then, each activity in the trace is mapped into a state, e.g., a and b in trace σ_1 are mapped to s_a and s_b, respectively. And the state mapped by the last activity is the final state, e.g., b in trace σ_1 is the last activity so that s_b is the final state of transition system TS_1. The states are changed in the transition system when the activities are observed in the trace. Hence, adding the edge between the adjacent states, which is labeled by the observed activity.

Definition 6 (trace-based transition system). Let A be a set of activities. $\sigma \in A^*$ is a trace of length n over A. The trace-based transition system of σ is a transition system $TS = (S, \partial_{set}(\sigma), T)$, where

(1) $S = \{s_x \mid x \in \partial_{set}(\sigma)\} \cup \{s_0\}$

(2) $T = \{(s_0, \sigma[1], s_{\sigma[1]})\} \cup \{(s_{\sigma[j]}, \sigma[j+1], s_{\sigma[j+1]}) | 1 \leq j \leq n-1\}$

(3) $S^{start} = \{s_0\}$, i.e., s_0 is the *initial state*

(4) $S^{end} = \{s^*_{\sigma[n]}\}$, i.e., $s^*_{\sigma[n]}$ is the *final state*

Here, $\sigma[i]$ refers to the ith element of trace σ, and $\partial_{set}(\sigma) = \{\sigma[1], \sigma[2], \ldots, \sigma[n]\}$ converts a sequence into a set.

In Figure 2, the state with an arrow line is the initial state, and the one with two circles is the final state.

Transition system TS_1 depicted in Figure 2(a) can be formalized as follows: $S_1 = \{s_0, s_a, s_b\}$, $S_1^{start} = \{s_0\}$, $S_1^{end} = \{s_b\}$, $\partial_{set}(\sigma_1) = \{a, b\}$, and $T_1 = \{(s_0, a, s_a), (s_a, b, s_b)\}$.

Transition system TS_2 depicted in Figure 2(b) can be formalized as follows: $S_2 = \{s_0, s_b, s_d\}$, $S_2^{start} = \{s_0\}$, $S_2^{end} = \{s_d\}$, $\partial_{set}(\sigma_2) = \{b, d\}$, and $T_2 = \{(s_0, b, s_b), (s_b, d, s_d)\}$.

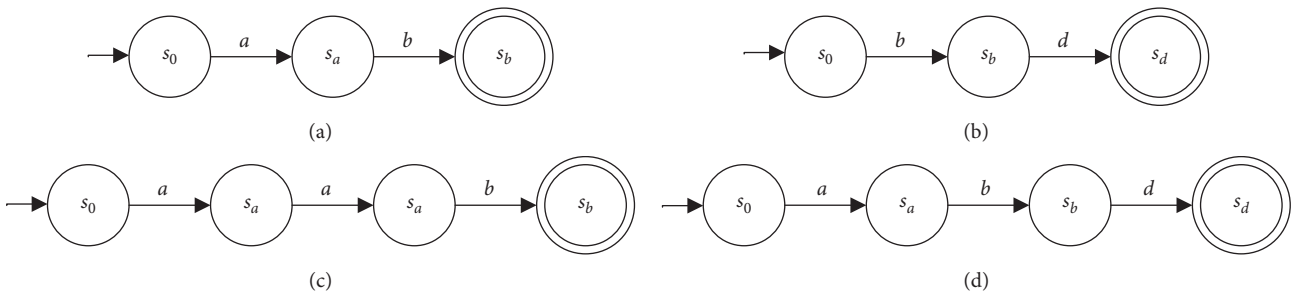

FIGURE 2: Transition systems of log L_1: (a) transition system TS_1 of σ_1; (b) transition system TS_2 of σ_2; (c) transition system TS_3 of σ_3; (d) transition system TS_4 of σ_4.

Transition system TS_3 depicted in Figure 2(c) can be formalized as follows: $S_3 = \{s_0, s_a, s_a, s_b\}$, $S_3^{\text{start}} = \{s_0\}$, $S_3^{\text{end}} = \{s_d\}$, $\partial_{\text{set}}(\sigma_3) = \{a, b\}$, and $T_3 = \{(s_0, a, s_a), (s_a, a, s_a), (s_a, b, s_b)\}$.

Transition system TS_4 depicted in Figure 2(d) can be formalized as follows: $S_4 = \{s_0, s_a, s_b, s_d\}$, $S_4^{\text{start}} = \{s_0\}$, $S_4^{\text{end}} = \{s_d\}$, $\partial_{\text{set}}(\sigma_4) = \{a, b, d\}$, and $T_4 = \{(s_0, a, s_a), (s_a, a, s_b), (s_b, d, s_d)\}$.

Through the analysis of the footprint in (6) and the transition systems in Figure 2, we can infer a relation matrix between states.

Definition 7 (log-based relation matrix). Let A be a set of activities. $L = [\sigma_1, \sigma_2, \ldots, \sigma_n] \subseteq A^*$ is an event log over A. $TS_i = (S_i, \partial_{\text{set}}(\sigma_i), T_i)$ is the trace-based transition system of trace σ_i, where $1 \le i \le n$. The log-based relation matrix is a matrix $LRM\ [|\bigcup_{i=1}^{n} \partial_{\text{set}}(\sigma_i)| + 1][|\bigcup_{i=1}^{n} \partial_{\text{set}}(\sigma_i)|]$, where

(1) $LRM^{\text{row}}\ [|\bigcup_{i=1}^{n} \partial_{\text{set}}(\sigma_i)| + 1] = \partial_{\text{set}}^{-1}(\{s_x \mid x \in \bigcup_{i=1}^{n} \partial_{\text{set}}(\sigma_i)\} \cup \{s_0\})^T$ is the *row mark* of LRM

(2) $LRM^{\text{col}}\ [|\bigcup_{i=1}^{n} \partial_{\text{set}}(\sigma_i)|] = \partial_{\text{set}}^{-1}(\{s_y \mid y \in \bigcup_{i=1}^{n} \partial_{\text{set}}(\sigma_i)\})$ is the *column mark* of LRM

(3) If $T = (LRM^{\text{row}}[j], a, LRM^{\text{col}}\ [k]) \in \bigcup_{i=1}^{n} \pi_3(TS_i)$, $LRM[LRM^{\text{row}}[j]][LRM^{\text{col}}[k]] = a$ and either $s_a = LRM^{\text{col}}[k]$ or $s_a^* = LRM^{\text{col}}[k]$; else, $LRM[LRM^{\text{row}}[j]][LRM^{\text{col}}[k]] = \#\,(1 \le j \le |\cup_{i=1}^{n} \partial_{\text{set}}(\sigma_i)|, 1 \le k \le |\cup_{i=1}^{n} \partial_{\text{set}}(\sigma_i)| - 1)$

(4) $LRM^{\text{start}} = \{s_0\}$, i.e., s_0 is the *initial state*

(5) $LRM^{\text{end}} = \{s_x^* \mid s_x^* \in S_i^{\text{end}}\}$, i.e., s_x^* is the *final state*

Here, $\partial_{\text{set}}^{-1}(S)$ is an inverse operation of $\partial_{\text{set}}(\sigma)$ and converts a set into a sequence. $\pi_i(x)$ refers to the ith element of x. The symbol # represents that the two states have no direct causal relation.

According Definition 7, the log-based relation matrix LRM_1 of log L_1 can be constructed, as in (7). The states labeled by the star in (7) are the final states:

$$\begin{array}{c} \\ s_0 \\ s_a \\ s_b^* \\ s_d^* \end{array} \begin{array}{c} \begin{array}{ccc} s_a & s_b^* & s_d^* \end{array} \\ \begin{bmatrix} a & b & \# \\ a & b & \# \\ \# & \# & d \\ \# & \# & \# \end{bmatrix} \end{array}. \tag{7}$$

According to Definition 7, the log-based relation matrix can present the sequences of activities. We read the state in the relation matrix, beginning at the initial state, and ending at the final state. A complete sequence consists of all the activities between states. The set including all the sequences is denoted as LRM^*. If LRM is the relation matrix of event log L, $L \subseteq LRM^*$, i.e., $\forall \sigma_i \in L$: $\sigma_i \subseteq LRM^*(1 \le i \le |L|)$. In contrast, $LRM^* \not\subseteq L$.

Obviously, $\sigma_1 \in LRM_1^*$, $\sigma_2 \in LRM_1^*$, $\sigma_3 \in LRM_1^*$, and $\sigma_4 \in LRM_1^*$, then $L_1 \subseteq LRM_1^*$. But $LRM_1^* \not\subseteq L_1$. Because there is the sequence $<a, a, a, a, b>$, where $<a, a, a, a, b> \in LRM_1^*$ but $<a, a, a, a, b> \notin L_1$.

4.2. Alignment Transition Systems. Given the process model $N = (P, T; F, \alpha, m_{\text{i}}, m_{\text{f}})$ and the event log $L = [\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_n]$, the relation matrix $LRM[S][S - \{s_0\}]$ is derived from the log firstly. We can measure the fitness between the log and model based on the relation matrix and Petri net.

From now on, we replace the event log as the relation matrix and use the matrix to embody the current state, observed activity, and next state of the log. Through the comparison between the relation matrix and Petri net, we can get an alignment transition system.

Definition 8 (alignment transition system). Let A be a set of activities. $L \subseteq A^*$ is an event log over A. $LRM[S][S - \{s_0\}]$ is the relation matrix of L. N=(P, T; F, α, m_{i}, m_{f}) is a Petri net over A. The *alignment transition system* between L and N is a transition system $ATS = (S, M, T)$, where

(1) $S = \{(m_k, s_x) \mid m_k \in R(m_i) \wedge s_x \in \partial_{\text{set}}(LRM^{\text{row}})\}$

(2) $T = \{(s_0, \sigma[1], s_{\sigma[1]})\} \cup \{(s_{\sigma[j]}, \sigma[j+1], s_{\sigma[j+1]}) \mid 1 \le j \le n-1\}$

(3) $S^{\text{start}} = \{(m_{\text{i}}, s_0)\}$ is the *initial state*

(4) $S^{\text{end}} = \{(m_f, s_y^*) \mid s_y^* \in \partial_{\text{set}}(LRM^{\text{row}})\}$ are the *final states*

We observe the log and run the process model. Meanwhile, we record the current state of the log and the current reachable marking of the model. Then, according to their current information, we infer that the activity can be observed in the matrix, the transition can be fired in the Petri net, and their relation can be compared. On the basis of the compared result, we can predict the next state of the matrix and the next reachable marking of the net.

For arbitrary Petri nets, their structures may be very complicated and diverse. Here, we discuss a special structure for the Petri net and its influence on the alignment transition

system. The Petri net contains cycles in which the cost of the transitions is 0. As a result, the alignment transition system may contain cycles with cost 0. In the context of practical application of our paper, the transitions with cost 0 are the invisible transitions in the Petri nets. The invisible transitions represent the activities that can never be observed, so the cycles containing only the invisible transitions have little meanings to the alignment results. In order to reach the final node from the initial node in a limited number of steps, we delete this kind of cycles from the alignment transition system.

The abovementioned idea is translated into the concrete algorithm to realize the alignment transition system, as shown in Algorithm 1.

The computational complexity of the alignment transition systems based Petri nets and relation matrices is related to the number of the reachable states of the Petri nets and the length of the traces, which is a NP-hard problem. Its complexity is also very high just like the reachable marking graph of the Petri nets. Especially, when there are many transitions with the concurrent relations in Petri nets, the number of the reachable states increases exponentially and even causes state space to explode.

According to Algorithm 1, taking relation matrix LRM_1 in (7) and process model N_1 in Figure 1 as an example, we can get an alignment transition system, as shown in Figure 3.

Given the event log and the process model, the alignment between them is translated into the calculation of the move sequence in the alignment transition system. The moves are the weights of the edges in the system, so we can calculate the move sequence when traversing the branch of the system. Hence, the problem of the optimal alignment between the log and the model is translated into that of the minimum cost move sequence in the alignment transition system. However, the alignment transition system includes all the information of the event log, so the sequence consisted of the projection of the moving sequence onto the first column must be the given trace when calculating the optimal alignments between the trace and the model.

When visiting the states in alignment transition system ATS_1 in Figure 3, we can get some shortest paths, as shown in Figure 4, and their corresponding optimal alignments, as shown in Figure 5.

4.3. Properties of Alignment Transition Systems. In this section, we present the properties of the alignment transition system and theoretically demonstrate that it includes the alignments between all traces in the event log and the process model.

Theorem 1. *Let $N = (P, T_N; F, \alpha, m_i, m_f)$ is a Petri net, L is an event log, LRM is a log-based relation matrix of L, and $ATS = (S_A, M_A, T_A)$ is an alignment transition system of LRM and N. $S_A = R(m_i) \times LRM^{row}$, where $R(m_i)$ is all the reachable states of N, and LRM^{row} includes all the states in LRM.*

Proof. $\forall s_A \in S_A$, $\pi_1(s_A) \in R(m_i) \wedge \pi_2(s_A) \in LRM^{\text{row}}$. So, $s_A \in R(m_i) \times LRM^{\text{row}}$. Then, $S_A \subseteq R(m_i) \times LRM^{\text{row}}$.

$\forall (m_j, s_{a_k}) \in R(m_i) \times LRM^{\text{row}}$, we examine whether a path can be established from (m_i, s_0) to (m_j, s_{a_k}) according to Algorithm 1. $m_j \in R(m_i)$, we suppose that a firing transition sequence $t_1, t_2 \ldots t_j$ in N, which makes $m_i[t_1\ t_2 \ldots t_j > m_j$. Then, there is a transition sequence in ATS which makes $(m_j, s_0) \in S_A$. The sequence is $<((m_i, s_0), (>>, t_1), (m_1, s_0)), ((m_1, s_0), (\gg, t_2), (m_2, s_0)), \ldots, ((m_j, s_0), (\gg, t_j), (m_j, s_0)) >\cdot s_{a_k} \in LRM^{\text{row}}$; we suppose that an activity sequence $a_1, a_2, \ldots, a_k$ in LRM^{row}, which makes $LRM[s_0][s_{a_1}] = a_1, LRM[s_{a_1}][s_{a_2}] = a_2, \ldots$, and $LRM[s_{a_{k-1}}][s_{a_k}] = a_k$. Then, there is a transition sequence in ATS which makes $(m_j, s_{a_k}) \in S_A$. The sequence is $<((m_j, s_0), (a_1, \gg), (m_j, s_{a_1}))$, $((m_j, s_{a_1}), (a_2, \gg), (m_j, s_{a_2}))$, $\ldots$, $((m_j, s_{a_{k-1}}), (a_k, \gg), (m_j, s_{a_k}))>$. Hence, $(m_j, s_{a_k}) \in S_A$. So, $R(m_i) \times LRM^{\text{row}} \subseteq S_A$.

In conclusion, $S_A = R(m_i) \times LRM^{\text{row}}$.

Theorem 1 shows that the state set of the alignment transition system is the Cartesian product of the reachable state set of the Petri net and the state set of the relation matrix. According to Theorem 1, no matter what states we use, they must be in alignment transition system.

For example, $R(m_{i,1})$ of N_1 in Figure 1 is $\{[p_1], [p_2], [p_3], [p_4]\}$ and LRM_1^{row} in (7) is $\{s_0, s_a, s_b, s_d\}$. S_{ATS1} of ATS_1 in Figure 3 is $\{([p_1], s_0), ([p_2], s_0), ([p_3], s_0), ([p_4], s_0), ([p_1], s_a), ([p_2], s_a), ([p_3], s_a), ([p_4], s_a), ([p_1], s_b), ([p_2], s_b), ([p_3], s_b), ([p_4], s_b), ([p_1], s_d), ([p_2], s_d), ([p_3], s_d), ([p_4], s_d)\}$, which is equal to $R(m_{i,1}) \times LRM_1^{\text{row}}$. □

Theorem 2. *Let $N = (P, T_N; F, \alpha, m_i, m_f)$ is a Petri net, L is an event log, LRM is a log-based relation matrix of L, and $ATS = (S_A, M_A, T_A)$ is an alignment transition system of LRM and N. $\Gamma_{L,N}$ is the set of all alignments between all traces in L and N. $\Gamma_{L,N} \subseteq ATS^*$, and ATS^* includes all the sequences that are $\pi_2(<(s_{A,1}, m_{A,1}, s_{A,2}), (s_{A,2}, m_{A,2}, s_{A,3}), \ldots, (s_{A,\ n-1}, m_{A,n}, s_{A,n})>)$, where $s_{A,1} \in S_A^{start}$ and $s_{A,n} \in S_A^{end}$.*

Proof. $\forall \gamma = \gamma[1], \gamma[2], \ldots, \gamma[q] > \in \Gamma_{L,N}$, ignoring $>>$, $\sigma = \pi_1(\gamma) \in L$ and $\lambda = \pi_2(\gamma) \in T_N^*$. $<\gamma[1]>, <\gamma[1], \gamma[2]>, \ldots, <\gamma[1], \gamma[2], \ldots, \gamma[i]>, \ldots$, and $<\gamma[1], \gamma[2], \ldots, \gamma[q]>$ are the prefix of γ, and the prefix alignments between L and N, where $1 \le i \le q$.

We prove the theorem by induction on $|\gamma|$, where $|\gamma| = q$:

(1) When $i = 1$, ① if $\gamma[1]$ is a log move, $\gamma[1] = (a_k, >>)$. a_k is the first activity of trace σ. So, $\exists s_{a_k} \in LRM^{\text{row}}$, $LRM[s_0][s_{a_k}] = a_k$. According to Steps 7–13 of Algorithm 1, $\exists((m_i, s_0), (a_k, >>), (m_i, s_{a_k}))$ is a transition of ATS. ② If $\gamma[1]$ is a model move, $\gamma[1] = (>>, t_j)$. t_j is the first transition of λ. $\exists m_j \in R(m_i)$, $m_i[t_j > m_j$. According to Step 14–Step 23 of Algorithm 1, $\exists((m_i, s_0), (>>, t_j), (m_j, s_0))$ is a transition of ATS. ③ If $\gamma[1]$ is a synchronous move, $\gamma[1] = (a_k, t_j)$. According to Step 24–Step 31 of Algorithm 1, $\exists((m_i, s_0), (a_k, t_j), (m_j, s_{a_k}))$ is a transition of ATS, where $m_i[t_j > m_j$. Hence, $<\gamma[1]>$ is the prefix of some sequences in ATS^*.

Input: Petri net model N_1=(P_1, T_1; F_1, α_1, $m_{i,1}$, $m_{f,1}$), and relation matrix LRM[S][$S - \{s_0\}$].
Output: alignment transition system $TS = (S, M, T)$.
Initialize: $S \longleftarrow \varnothing$, $M \longleftarrow \varnothing$, $T \longleftarrow \varnothing$, $S^{\text{start}} \longleftarrow \{(m_{i,1}, s_0)\}$, $S^{\text{end}} \longleftarrow \varnothing$.
(1) $S \longleftarrow S^{\text{start}}$; $n \longleftarrow 1$;
(2) WHILE ($n \le |S|$) DO
(3) $(m_j, s_x) \longleftarrow S[n]$;
(4) IF($m_j = m_{f,1}$ AND $s_x = s_x^* \in \partial_{\text{set}}(\text{LRM}^{\text{row}})$) THEN
(5) $S^{\text{end}} \longleftarrow S^{\text{end}} \cup \{(m_j, s_x^*)\}$;
(6) END IF
//*judge the current state to be the final state;*
(7) FOR (all $s_y \in \partial_{\text{set}}(\text{LRM}^{\text{row}})$) DO
(8) IF (LRM[s_x][s_y] $\neq$ #) THEN
(9) $M \longleftarrow M \cup \{(\text{LRM}[s_x][s_y], >>)\}$;
(10) $S \longleftarrow S \cup \{(m_j, s_y)\}$;
(11) $T \longleftarrow T \cup \{(m_j, s_x), (\text{LRM}[s_x][s_y], >>), (m_j, s_y)\}$;
(12) END IF
(13) END FOR
//the following log moves, related new states, and transitions that may be generated;
(14) IF($m_j \neq m_{f,1}$) THEN
(15) FOR(all $t_k \in T_1$) DO
(16) IF($^{\cdot}t_k \in m_j$) THEN
(17) $M \longleftarrow M \cup \{(>>, t_k)\}$;
(18) $m_j[t_k > m_y$;
(19) $S \longleftarrow S \cup \{(m_y, s_x)\}$;
(20) $T \longleftarrow T \cup \{(m_j, s_x), (>>, t_k), (m_y, s_x)\}$;
(21) END IF
(22) END FOR
(23) END IF
//the following model moves, related new states, and transitions that may be generated;
(24) FOR((all $s_y \in \partial_{\text{set}}(\text{LRM}^{\text{row}})$) AND (all $t_k \in T_1$)) DO
(25) IF ($^{\cdot}t_k \in m_j$) AND (LRM[s_x][s_y] = $\alpha(t_k)$) THEN
(26) $M \longleftarrow M \cup \{(\alpha(t_k), t_k)\}$;
(27) $m_j\ [t_k > m_y$;
(28) $S \longleftarrow S \cup \{(m_y, s_y)\}$;
(29) $T \longleftarrow T \cup \{(m_j, s_x), (\alpha(t_k), t_k), (m_y, s_y)\}$;
(30) END IF
(31) END FOR
//the following synchronous moves, related new states, and transitions that may be generated;
(32) $n \longleftarrow n + 1$;
(33) END WHILE
//delete the cycles with cost 0 in the transition system;
(34) FOR (all cycles with cost 0 in TS) Do
(35) Delete all the edges with cost 0;
(36) FOR(all nodes in the cycle) DO
(37) FOR(all *nodes* have no out edge) DO
(38) Delete *nodes*;
(39) Set the parents of nodes to be *nodes*;
(40) END FOR
(41) END FOR
(42) END FOR
(43) RETURN TS;

Algorithm 1: The generation algorithm of the alignment transition system between the event log and the process model.

(2) When $i = q - 1$, it is supposed that $<\gamma[1], \gamma[2], \ldots, \gamma[q-1]>$ is also the prefix of some sequences in ATS^*. We suppose the last state in ATS is (m_j, s_{a_k}).

Let $i = q$, ① if $\gamma[q]$ is a log move, $\gamma[q] = (a_{k+1}, >>)$. a_{k+1} is the last activity of trace σ and $m_j = m_f$. So, $\exists s_{a_{k+1}} \in LRM^{\text{row}} s_{a_{k+1}}^{\wedge} \in LRM^{\text{end}}$, $LRM[s_{a_k}][s_{a_{k+1}}] = a_{k+1}$. According to Steps 7–13 of Algorithm 1, $\exists((m_f, s_{a_k}), (a_{k+1}, \gg), (m_f, s_{a_{k+1}}))$ is a transition of ATS. ② If $\gamma[q]$ is a model move, $\gamma[q] = (>>, t_{j+1})$. t_{j+1} is the last transition of λ and $s_{a_k} \in LRM^{\text{end}}$. $\exists m_{j+1} \in R(m_i) \wedge m_{j+1} = m_f$, $m_j[t_{j+1} > m_{j+1}$. According to Steps 14–23 of Algorithm 1, $\exists((m_j, s_{a_k}), (\gg, t_{j+1}), (m_j, s_{a_k}))$ is a transition of ATS. ③ If $\gamma[q]$ is a synchronous move, $\gamma[q] = (a_{k+1}, t_{j+1})$. According to Step

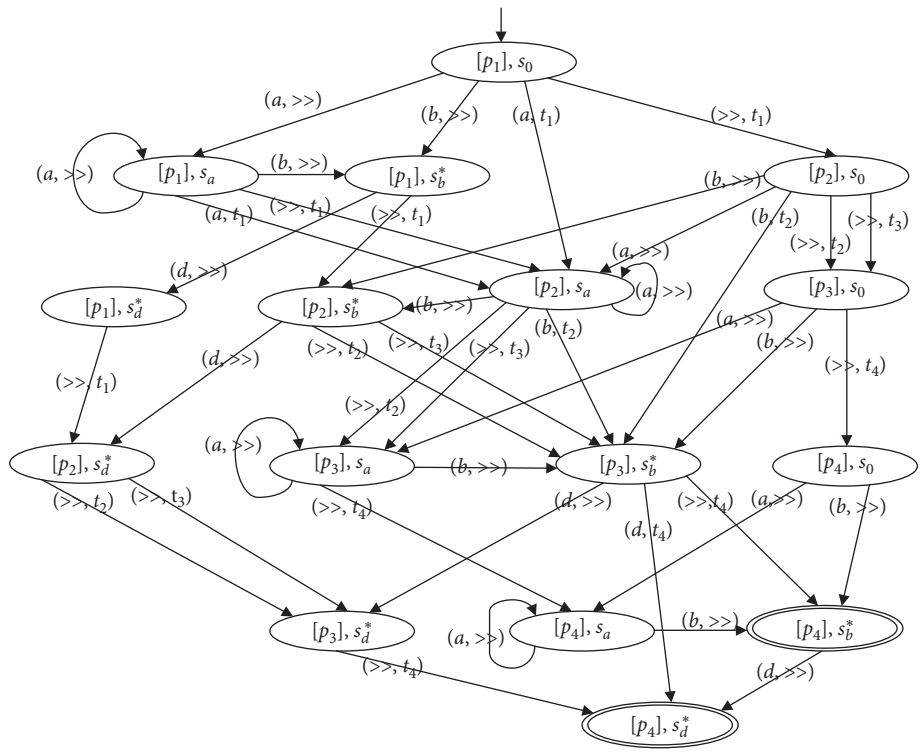

Figure 3: Alignment transition system ATS_1.

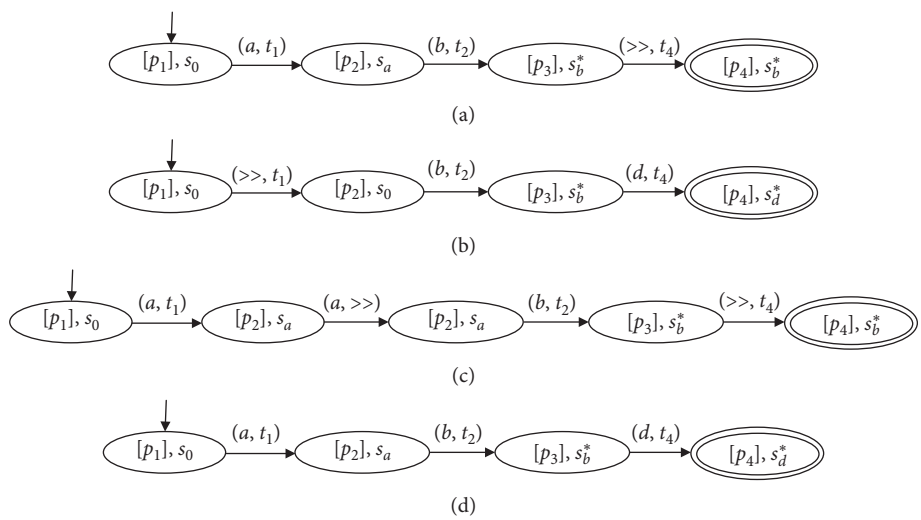

Figure 4: Paths from the initial state to the final states mapping to the optimal alignments of traces in L_1: (a) a shortest path for the optimal alignment of σ_1; (b) a shortest path for the optimal alignment of σ_2; (c) a shortest path for the optimal alignment of σ_3; (d) a shortest path for the optimal alignment of σ_4.

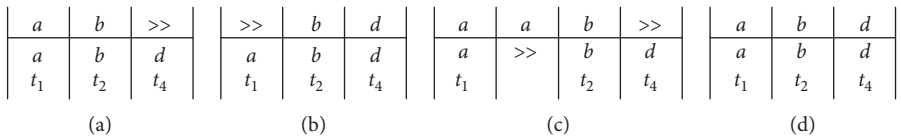

Figure 5: Optimal alignments of traces in L_1: (a) an optimal alignment of σ_1; (b) an optimal alignment of σ_2; (c) an optimal alignment of σ_3; (d) an optimal alignment of σ_4.

24–31 of Algorithm 1, $\exists((m_j, s_{a_k}), (a_{k+1}, t_{j+1}), (m_f, s_{a_{k+1}}))$ is a transition of ATS. Hence, $<\gamma[1], \gamma[2], \ldots, \gamma[q]>$ is a sequence in ATS^*.

In conclusion, $\gamma \in ATS^*$. Then, $\Gamma_{L,N} \subseteq ATS^*$.

Theorem 2 shows that all the alignments between the traces in the log and the model can be found in the alignment transition system. □

Corollary 1. *Let $N = (P, T_N; F, \alpha, m_i, m_f)$ is a Petri net, L is an event log, LRM is a log-based relation matrix of L, and $ATS = (S_A, M_A, T_A)$ is an alignment transition system of LRM and N. $\Gamma^o_{L,N,lc}$ is the set of all the optimal alignments between the traces in L and N based on the standard likelihood cost function $lc()$. $\Gamma^o_{L,N,lc} \subseteq ATS^*$.*

Proof. According to Definition 5, $\Gamma^o_{L,N,lc} \subseteq \Gamma_{L,N}$. According to Theorem 2, $\Gamma_{L,N} \subseteq ATS^*$.

Hence, $\Gamma^o_{L,N,lc} \subseteq ATS^*$.

Corollary 1 shows that all the optimal alignments between the traces in the log and the model can be found in the alignment transition system. This corollary provides the theoretical foundation for the search of the optimal alignments in the alignment transition system. □

5. Calculation of Optimal Alignments

We can get an alignment transition system between the event log and the process model by Algorithm 1. The optimal alignments between the trace and model based on the standard likelihood cost function can be calculated through finding the shortest path from the initial state to the final state in the system. This section presents two algorithms to realize the calculations of an optimal alignment and all optimal alignments, respectively.

Since most of the states in the alignment transition system have more than one parent and even some states have self-cycles, it is possible to reach the same state from different branches in the process of finding the optimal alignment. In the search process, we should record not only the current state and cost but also the prefix alignment. The unit that stores the related information is referred to as the search node.

5.1. Computing an Optimal Alignment. Because there are more than one trace in the event log, a path between the initial state and the final state in the alignment transition system may not be the alignment of the given trace, let alone its optimal alignment. In other words, the shortest path may not be mapping to the optimal alignment for the specified trace. So, not only the length of the path but also the specified trace must be considered when computing the optimal alignment between the trace and the model.

In order to get the optimal alignment of the given trace, not only the short path should be searched as far as possible but also it should be guaranteed that the sequence consisted of the projection onto the first column of the moves on the path is equal to the prefix of the trace. Algorithm 2 is presented to compute an optimal alignment between all traces in the event log and the process model based on the standard likelihood cost function.

The complexity of Algorithm 2 is related to that of the alignment transition system and the given trace, which is a NP-hard problem. In Algorithm 2, each node stores the prefix alignment. When calculating the optimal alignment, it must meet the two following criteria: one is that the last visited state of the alignment transition system must be the final state; the other is that the projection of the eventual prefix alignment onto the first column is the given trace. Most of the nodes generated in the search process can be discarded, only the current node needs to be stored, so the storage space of this algorithm is greatly saved.

According to Algorithm 2, taking alignment transition system ATS_1 in Figure 3 and event log L_1 in Table 1 as an example, we can get an optimal alignment for each trace in log L_1. When visiting the states in alignment transition system ATS_1 in Figure 3, the key nodes generated, as shown in Table 2. The prefix alignment of the last node for each trace is its optimal alignment. The search results of each step in Algorithm 2 can ensure that the prefix alignment is with the least cost at the current cost, and it is suitable for the given trace. The eventual results are certain to be the optimal alignments of the given traces due to the current costs and the prefix alignments.

5.2. Computing all Optimal Alignments. The alignment transition system includes all the alignments between all the traces and the model. When calculating the optimal alignments based on the system, we can get more alignments besides the optimal ones if ignoring the constraint of the least cost. In fact, when having found the optimal alignment for the first time, its cost is the minimum value between the given trace and the model based on the standard likelihood cost function. The cost of any other optimal alignments cannot be greater than this value.

In addition, all the nodes whose cost is less than or equal to the cost of the optimal alignment are checked, which can determine whether we get all optimal alignment. However, if the cost of the node is greater than the optimal cost, it will never arrive at the final node which stands for the optimal alignment.

The main idea to compute all optimal alignment is similar to that of Algorithm 2. The successor of the current node can be entered into the queue when it meets the two criteria: one is that the current cost is not greater than the optimal cost; the other is that the projection of the current alignment onto the first column conforms to the prefix of the given trace.

Algorithm 3 is presented to achieve all optimal alignments between all traces in the event log and the process model based on standard likelihood cost function.

This algorithm can be further optimized. For instance, if the successor is equal to the existing node, we can abort to add the successor and share the existing one. After optimizing the algorithm, the number of the nodes in the queue will be reduced, so the efficiency of the algorithm can be improved.

Input: event log $L_1 = [\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_n]$, and alignment transition system $TS_2 = (S_2, M_2, T_2)$ between L_1 and $N_1 = (P_1, T_1; F_1, \alpha_1, m_{i,1}, m_{f,1})$.
Output: $OA\_one[n]$($OA\_one[i]$ stores an optimal alignment γ_i between the ith trace and the model, where $1 \le i \le n$).

(1) FOR(all $\sigma_i \in L_1$) DO
(2) *queue* ⟵ ∅;
(3) *firststate* ⟵ ($m_{i,1}$, s_0);
(4) *firstalign* ⟵ <>;
(5) *firstcost* ⟵ 0;
(6) *firstnode* ⟵ (*firststate*, *firstalign*, *firstcost*);
//add the initial node to the queue;
(7) *queue* ⟵ *queue* ∪ {*firstnode*};
(8) WHILE(*queue* ≠ ∅) DO
(9) FOR (all *node* ∈ *queue*) Do
(10) choose the *node* with the minimum cost as *curnode*;
(11) *queue* ⟵ *queue* − {*curnode*};
(12) END FOR
//delete the current node from the queue;
(13) FOR(all $t_j \in T_2$) DO
(14) IF($\pi_1(t_j) = \pi_1(curnode)$) THEN
(15) $\gamma_i = \pi_2(curnode) \oplus <(\pi_2(t_j)>$;
(16) IF $(\pi_1(\gamma_i)_{\downarrow \cup_{i=1}^{n} \partial_{set}(\sigma_i)})$ is the prefix of σ_i THEN
(17) IF $(\pi_1(\gamma_i)_{\downarrow \cup_{i=1}^{n} \partial_{set}(\sigma_i)} = \sigma_i)$ AND $(\pi_3(t_j) \in S_2^{end})$THEN
(18) $OA\_one[i]$ ⟵ γ_i;
(19) JUMP TO Step 1;
//obtain the optimal alignment for the given trace and be ready to do the search for the next one;
(20) ELSE
(21) *sucstate* ⟵ $\pi_3(t_j)$;
(22) *sucalign* ⟵ γ_i;
(23) *succost* ⟵ $\pi_3(curnode) + lc(\pi_2(t_j))$;
(24) *sucnode* ⟵ (*sucstate*,*sucalign*, *succost*);
(25) *queue* ⟵ *queue* ∪ {*sucnode*};
//add the successor to the queue;
(26) END IF
(27) END IF
(28) END IF
(29) END FOR
(30) END WHILE
(31) END FOR
(32) RETURN *OA_one*;

Algorithm 2: Computing an optimal alignment between the event log and the process model based on the standard likelihood cost function.

The execution of Algorithm 3 mentions to the traverse of the alignment transition system, and some states in the system will be visited for several times. Hence, its time complexity is very high. The complexity of Algorithm 3 is still a NP-hard problem.

According to Algorithm 3, taking alignment transition system ATS_1 in Figure 3 and event log L_1 in Table 1 as an example, we can get all optimal alignments for each trace in log L_1. σ_1, σ_2, and σ_4 have just only one optimal alignment. However, trace σ_3 has two optimal alignments, as shown in Figure 6.

6. Scalability of Our Approach

In most of the cases, the existing approaches can only compute the alignment between one trace and the model each time. When computing the optimal alignments between a new trace and the model, we must completely execute the alignment approach again and get a different search space. However, the proposed approach in this paper has certain scalability. We just adjust and modify the alignment transition system, and then it can be used for the new trace.

Next, we introduce the possible changes for the alignment transition system when aligning a new trace with the model.

6.1. Remaining Unchanged. When the relations of the activities in the new trace conform to that of the original log-based relation matrix and the last activity is identical to that of some existing traces, the alignment transition system remains unchanged.

Table 2: The key nodes for each trace of $\log L_1$ in Algorithm 2.

Trace	Node	Current state	Prefix alignment	Current cost
σ_1	v_1	$([p_1], s_0)$	<>	0
	v_2	$([p_2], s_a)$	$<(a, t_1)>$	0
	v_3	$([p_3], s_b^*)$	$<(a, t_1), (b, t_2)>$	0
	v_4	$([p_4], s_b^*)$	$<(a, t_1), (b, t_2), (>>, t_4)>$	1
σ_2	v_1	$([p_1], s_0)$	<>	0
	v_2	$([p_2], s_0)$	$<(>>, t_1)>$	1
	v_3	$([p_3], s_b^*)$	$<(>>, t_1), (b, t_2)>$	1
	v_4	$([p_4], s_d^*)$	$<(>>, t_1), (b, t_2), (d, t_4)>$	1
σ_3	v_1	$([p_1], s_0)$	<>	0
	v_2	$([p_2], s_a)$	$<(a, t_1)>$	0
	v_3	$([p_2], s_a)$	$<(a, t_1), (a, >>)>$	1
	v_4	$([p_3], s_b^*)$	$<(a, t_1), (a, >>), (b, t_2)>$	1
	v_5	$([p_4], s_b^*)$	$<(a, t_1), (a, >>), (b, t_2), (>>, t_4)>$	2
σ_4	v_1	$([p_1], s_0)$	<>	0
	v_2	$([p_2], s_a)$	$<(a, t_1)>$	0
	v_3	$([p_3], s_b^*)$	$<(a, t_1), (b, t_2)>$	0
	v_4	$([p_4], s_b^*)$	$<(a, t_1), (b, t_2), (d, t_4)>$	0

Taking process model N_1 in Figure 1, event log L_1 in Table 1, and alignment transition system ATS_1 in Figure 3 as an example, the log-based relation matrix of L_1 is LRM_1 in (7). We suppose that the new trace is $\sigma_5 = <a, a, a, b>$, which is different from any trace in L_1. And its trace-based transition system is shown in Figure 7.

Transition system TS_5 depicted in Figure 7 can be formalized as follows: $S_5 = \{s_0, s_a, s_b^*\}$, $S_5^{\text{start}} = \{s_0\}$, $S_5^{\text{end}} = \{s_b^*\}$, $\partial_{\text{set}}(\sigma_5) = \{a, b\}$, and $T_5 = \{(s_0, a, s_a), (s_a, a, s_a), (s_a, b, a_b^*)\}$. Compared with LRM_1, $S_5 \subseteq LRM_1^{\text{row}}$, $S_5 - \{s_0\} \subseteq LRM_1^{\text{col}}$, $S_5^{\text{start}} \subseteq LRM_1^{\text{start}}$, and $S_5^{\text{end}} \subseteq LRM_1^{\text{end}}$. And T_5 can be depicted by LRM_1 as follows: $LRM_1[s_0][s_a] = a$, $LRM_1[s_a][s_a] = a$, and $LRM_1[s_a][s_b^*] = b$. Hence, we can align trace σ_5 with model N_1 by system ATS_1.

We can get an optimal alignment between trace σ_5 and model N_1 by Algorithm 2 from transition system ATS_1 and all the optimal alignments by Algorithm 3. The optimal alignments are corresponding to the paths from the initial node to the final nodes in ATS_1, as shown in Table 3.

6.2. Setting Another New Final State. When the relations of the activities in the new trace conform to that of the original log-based relation matrix, but the last activity is different from that of any existing trace, the alignment transition system needs to set another new final state. Supposing that the last activity is x, we must set the state (m_f, s_x^*) to be another new final state in the alignment transition system.

We suppose the new trace is $\sigma_6 = <a, a>$, which is different from any trace in L_1. And its trace-based transition system is shown in Figure 8.

Transition system TS_6 depicted in Figure 8 can be formalized as follows: $S_6 = \{s_0, s_a\}$, $S_6^{\text{start}} = \{s_0\}$, $S_6^{\text{end}} = \{s_a\}$, $\partial_{\text{set}}(\sigma_6) = \{a\}$, and $T_6 = \{(s_0, a, s_a), (s_a, a, s_a)\}$. Compared with LRM_1, $S_6 \subseteq LRM_1^{\text{row}}$, $S_6 - \{s_0\} \subseteq LRM_1^{\text{col}}$, and $S_6^{\text{start}} \subseteq LRM_1^{\text{start}}$. And T_6 can be depicted by LRM_1 as follows: $LRM_1[s_0][s_a] = a$ and $LRM_1[s_a][s_a] = a$. But $S_6^{\text{end}} \notin LRM_1^{\text{end}}$. Hence, we must set s_a to be another new final state s_a^* in LRM_1 as follows:

$$\begin{array}{c} \\ s_0 \\ s_a^* \\ s_b^* \\ s_d^* \end{array} \begin{array}{c} \begin{array}{ccc} s_a^* & s_b^* & s_d^* \end{array} \\ \begin{bmatrix} a & b & \# \\ a & b & \# \\ \# & \# & d \\ \# & \# & \# \end{bmatrix} \end{array}. \tag{8}$$

Accordingly, we set the state $([p_4], s_a^*)$ to be another new final state in the alignment transition system. Supposing that the modified system is named as ATS_1^1, we can align trace σ_6 with model N_1 by system ATS_1^1. We can get an optimal alignment between trace σ_6 and model N_1 by Algorithm 2 from transition system ATS_1^1 and all the optimal alignments by Algorithm 3. The optimal alignments are corresponding to the paths from the initial node to the final nodes in ATS_1^1, as shown in Table 4.

6.3. Adding New Transitions. When the relations of the activities in the new trace cannot be found in the original log-based relation matrix, the alignment transition system needs to add new transitions. We suppose that the new relation is $x \longrightarrow_L y$, then we must add new transitions to the alignment transition system. The new transitions include $\{((m_k, s_x), (y, >>), (m_k, s_y)) | m_k \in R(m_i) \wedge (m_k, s_x) \in ATS\} \cup \{((m_k, s_x), (y, t_j), (m_{k+1}, s_y)) | m_k \in R(m_i) \wedge (m_k, s_x) \in ATS \wedge m_k[t_j > m_{k+1} \wedge \alpha(t_j) = y\}$.

We suppose the new trace is $\sigma_7 = <b, a, b>$, which is different from any trace in L_1. And its trace-based transition system is shown in Figure 9.

Transition system TS_7 depicted in Figure 9 can be formalized as follows: $S_7 = \{s_0, s_a, s_b^*\}$, $S_7^{\text{start}} = \{s_0\}$, $S_7^{\text{end}} = \{s_b^*\}$, $\partial_{\text{set}}(\sigma_7) = \{a, b\}$, and $T_7 = \{(s_0, b, s_b^*), (s_b^*, a, s_a), (s_a, b, s_b^*)\}$. Compared with LRM_1, $S_7 \subseteq LRM_1^{\text{row}}$, $S_7 - \{s_0\} \subseteq LRM_1^{\text{col}}$, $S_7^{\text{start}} \subseteq LRM_1^{\text{start}}$, and $S_7^{\text{end}} \subseteq LRM_1^{\text{end}}$. And T_7 can partly be depicted by LRM_1 as follows: $LRM_1[s_0][s_b^*] = b$ and $LRM_1[s_a][s_b^*] = a$. But transition (s_b^*, a, s_a) in T_7 cannot be expressed in LRM_1. Firstly, we set $LRM_1[s_b^*][s_a] = a$ as follows:

Input: event log $L_1 = [\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_n]$, and alignment transition system $TS_2 = (S_2, M_2, T_2)$ between L_1 and $N_1 = (P_1, T_1; F_1, \alpha_1, m_{i,1}, m_{f,1})$.
Output: $OA\_all[n]$ ($OA\_all[i]$ stores all optimal alignments $\Gamma^{o}_{\sigma_i, N_1, lc}$ between the ith trace and the model, where $1 \le i \le n$).

(1) FOR(all $\sigma_i \in L_1$) DO
(2) $queue \longleftarrow \varnothing$;
(3) $cost \longleftarrow +\infty$;
(4) $firststate \longleftarrow (m_{i,1}, s_0)$;
(5) $firstalign \longleftarrow <>$;
(6) $firstcost \longleftarrow 0$;
(7) $firstnode \longleftarrow (firststate, firstalign, firstcost)$;
//add the initial node to the queue;
(8) $queue \longleftarrow queue \cup \{firstnode\}$;
(9) WHILE($queue \neq \varnothing$) DO
(10) FOR(all $node \in queue$) Do
(11) choose $node$ with the minimum cost as $curnode$;
(12) $queue \longleftarrow queue - \{curnode\}$;
(13) END FOR
//delete the current node from the queue;
(14) FOR(all $t_j \in T_2$) DO
(15) IF ($\pi_1(t_j) = \pi_1(curnode)$) THEN
(16) $\gamma_i = \pi_2(curnode) \oplus <(\pi_2(t_j)>$;
(17) $c_i = \pi_3(curnode) + lc(\pi_2(t_j))$;
(18) IF($c_i > cost$) THEN
(19) CONTINUE;
(20) ELSE
(21) IF $\pi_1(\gamma_i)_{\downarrow \cup_{i=1}^{n} \partial_{set}(\sigma_i)}$ is σ_i' prefix THEN
(22) IF($\pi_1(\gamma_i)_{\downarrow \cup_{i=1}^{n} \partial_{set}(\sigma_i)} = \sigma_i$ AND ($\pi_3(t_j) \in S_2^{\text{end}}$)) THEN
(23) $cost = c_i$;
//update the minimum cost;
(24) $OA\_all[i] \longleftarrow OA\_all[i] \cup \gamma_i$;
//obtain an optimal alignment for the ith trace;
(25) ELSE
(26) $sucstate \longleftarrow \pi_3(t_j)$;
(27) $sucalign \longleftarrow \gamma_i$;
(28) $succost \longleftarrow c_i$;
(29) $sucnode \longleftarrow (sucstate, sucalign, succost)$;
(30) $queue \longleftarrow queue \cup \{sucnode\}$;
//add the successor to the queue;
(31) END IF
(32) END IF
(33) END IF
(34) END IF
(35) END FOR
(36) END WHILE
(37) END FOR
//*find all the optimal alignments for trace* σ_i;
(38) RETURN OA_all;

Algorithm 3: Computing all optimal alignments between the event log and the process model based on the standard likelihood cost function.

a	a	b	>>
a	>>	b	d
t_1		t_2	t_4

(a)

a	a	b	>>
>>	a	b	d
	t_1	t_2	t_4

(b)

Figure 6: All optimal alignments of trace σ_3: (a) the optimal alignment γ_{31}; (b) the optimal alignment γ_{32}.

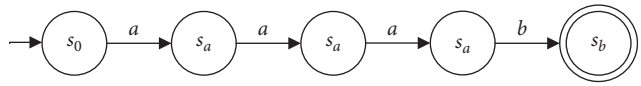

Figure 7: Transition system TS_5 of trace σ_5.

Table 3: Optimal alignments between trace σ_5 and model N_1.

Name	Path in ATS_1	Value
γ_{51}	$<(([p_1], s_0), (a, t_1), ([p_2], s_a)), (([p_2], s_a), (a, >>), ([p_2], s_a)), (([p_2], s_a), (a, >>), ([p_2], s_a)), (([p_2], s_a), (b, t_2), ([p_3], s_b^*)), ([p_3], s_b^*), (>>, t_4), ([p_4], s_b^*))>$	$<(a, t_1), (a, >>), (a, >>), (b, t_2), (>>, t_4)>$
γ_{52}	$<(([p_1], s_0), (a, >>), ([p_1], s_a)), (([p_1], s_a), (a, t_1), ([p_2], s_a)), (([p_2], s_a), (a, >>), ([p_2], s_a)), (([p_2], s_a), (b, t_2), ([p_3], s_b^*)), ([p_3], s_b^*), (>>, t_4), ([p_4], s_b^*))>$	$<(a, >>), (a, t_1), (a, >>), (b, t_2), (>>, t_4)>$
γ_{53}	$<(([p_1], s_0), (a, >>), ([p_1], s_a)), (([p_1], s_a), (a, >>), ([p_1], s_a)), (([p_1], s_a), (a, t_1), ([p_2], s_a)), (([p_2], s_a), (b, t_2), ([p_3], s_b^*)), ([p_3], s_b^*), (>>, t_4), ([p_4], s_b^*))>$	$<(a, >>), (a, >>), (a, t_1), (b, t_2), (>>, t_4)>$

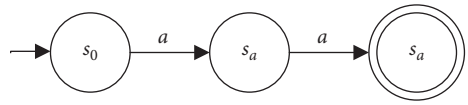

Figure 8: Transition system TS_6 of trace σ_6.

Table 4: Optimal alignments between trace σ_6 and model N_1.

Name	Path in ATS_1^1	Value
γ_{61}	$<(([p_1], s_0), (a, t_1), ([p_2], s_a)), (([p_2], s_a), (a,>>), ([p_2], s_a)), (([p_2], s_a), (>>, t_2), ([p_3], s_a)), (([p_3], s_a), (>>, t_4), ([p_4], s_a^*))>$	$<(a, t_1), (a,>>), (>>, t_2), (>>, t_4)>$
γ_{62}	$<(([p_1], s_0), (a,>>), ([p_1], s_a)), (([p_1], s_a), (a, t_1), ([p_2], s_a)), (([p_2], s_a), (>>, t_2), ([p_3], s_a)), (([p_3], s_a), (>>, t_4), ([p_4], s_a^*))>$	$<(a,>>), (a, t_1), (>>, t_2), (>>, t_4)>$
γ_{63}	$<(([p_1], s_0), (a, t_1), ([p_2], s_a)), (([p_2], s_a), (a,>>), ([p_2], s_a)), (([p_2], s_a), (>>, t_3), ([p_3], s_a)), (([p_3], s_a), (>>, t_4), ([p_4], s_a^*))>$	$<(a, t_1), (a,>>), (>>, t_3), (>>, t_4)>$
γ_{64}	$<(([p_1], s_0), (a,>>), ([p_1], s_a)), (([p_1], s_a), (a, t_1), ([p_2], s_a)), (([p_2], s_a), (>>, t_3), ([p_3], s_a)), (([p_3], s_a), (>>, t_4), ([p_4], s_a^*))>$	$<(a,>>), (a, t_1), (>>, t_3), (>>, t_4)>$

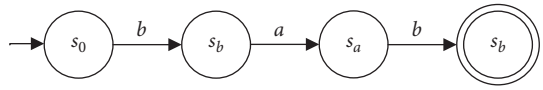

Figure 9: Transition system TS_7 of trace σ_7.

$$\begin{array}{c} \\ s_0 \\ s_a \\ s_b^* \\ s_d^* \end{array} \begin{array}{c} \begin{array}{cccc} s_a & s_b^* & s_d^* \end{array} \\ \begin{bmatrix} a & b & \# \\ a & b & \# \\ a & \# & d \\ \# & \# & \# \end{bmatrix} \end{array}. \tag{9}$$

Then, we add new transitions to ATS_1, including $\{((m_k, s_b^*), (a, \gg), (m_k, s_a)) \mid m_k \in \{[p_1], [p_2], [p_3], [p_4]\}\}$ and $(([p_1], s_b^*), (a, t_1), ([p_2], s_a))$. Supposing that the modified system is named as ${ATS_1}^2$, we can align trace σ_7 with model N_1 by system ATS_1^2. The adding transitions to ATS_1 are shown in Figure 10.

We can get an optimal alignment between trace σ_7 and model N_1 by Algorithm 2 from transition system ${ATS_1}^2$ and all the optimal alignments by Algorithm 3. The optimal alignments are corresponding to the paths from the initial node to the final nodes in ATS_1^2, as shown in Table 5.

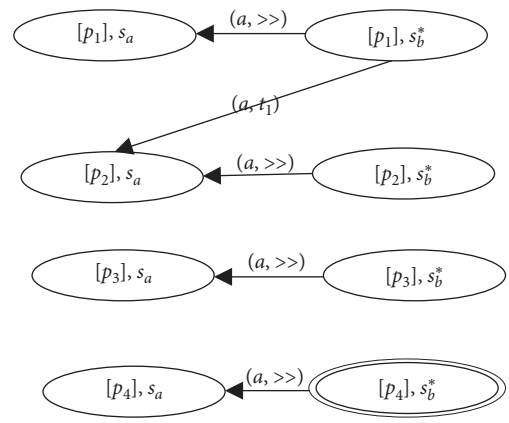

Figure 10: The newly added transitions in ATS_1^2 compared with ATS_1.

Table 5: Optimal alignments between trace σ_7 and model N_1.

Name	Path in ATS_1^2	Value
γ_{71}	$<(([p_1], s_0), (b, \gg), ([p_1], s_b^*)), (([p_1], s_b^*), (a, t_1), ([p_2], s_a)), (([p_2], s_2), (b, t_2), ([p_3], [s_b^*])), (([p_3], s_b^*), (\gg, t_4), ([p_4], s_b^*))>$	$<(b, >>), (a, t_1), (b, t_2), (>>, t_4)>$

6.4. Adding New States and Transitions. When the activities in the new trace cannot be found in the event log, the alignment transition system needs to add new states and transitions. Supposing that the activity is x, we must add several new states and transitions to the alignment transition system. The new states are as follows: $\{(m_k, s_x) | m_k \in R(m_i)\}$. The pattern that the new transitions are added is similar to the example in Section 6.3. We suppose the new trace is $\sigma_8 = <a, c, b>$, which is different from any trace in L_1. And its trace-based transition system is shown in Figure 11.

Transition system TS_8 depicted in Figure 11 can be formalized as follows: $S_8 = \{s_0, s_a, s_c, s_b^*\}$, $S_8^{\text{start}} = \{s_0\}$, $S_8^{\text{end}} = \{s_b^*\}$, $\partial_{\text{set}}(\sigma_8) = \{a, b, c\}$, and $T_8 = \{(s_0, a, s_a), (s_a, c, s_c), (s_c, b, s_b^*)\}$. Compared with LRM_1, $S_8^{\text{start}} \subseteq LRM_1^{\text{start}}$ and $S_8^{\text{end}} \subseteq LRM_1^{\text{end}}$. But, $S_8 \notin LRM_1^{\text{row}}$ and $S_8 - \{s_0\} \notin LRM_1^{\text{col}}$. T_8 can partly be depicted by LRM_1 as follows: $LRM_1[s_0][s_a] = a$. But transitions (s_a, c, s_c) and (s_c, b, s_b^*) in T_8 cannot be expressed in LRM_1. Firstly, we set $LRM_1[s_a][s_c] = c$ and $LRM_1[s_c][s_b^*] = b$, as follows:

$$\begin{array}{c} \\ s_0 \\ s_a \\ s_b^* \\ s_b^* \\ s_c \end{array} \begin{array}{c} \begin{array}{cccc} s_a & s_b^* & s_d^* & s_c \end{array} \\ \begin{bmatrix} a & b & \# & \# \\ a & b & \# & c \\ \# & \# & d & \# \\ \# & \# & \# & \# \\ \# & b & \# & \# \end{bmatrix} \end{array}. \tag{10}$$

Then, we add new states to ATS_1, including $\{(m_k, s_c) | m_k \in \{[p_1], [p_2], [p_3], [p_4]\}\}$. Next, we add new transitions, including $\{((m_k, s_a), (c, \gg), (m_k, s_c)) | m_k \in \{[p_1], [p_2], [p_3], [p_4]\}\}$, $\{((m_k, s_c), (b, \gg), (m_k, s_b^*)) | m_k \in \{[p_1], [p_2], [p_3], [p_4]\}\}$, $(([p_2], s_c), (b, t_2), ([p_3], s_b^*))$ and $(([p_2], s_a), (c, t_3), ([p_3], s_c))$. We suppose that the modified system is named as ${ATS_1}^3$, and the adding states and transitions to ATS_1 are shown in Figure 12.

We can align trace σ_8 with model N_1 by system ATS_1^3. We can get an optimal alignment between trace σ_8 and model N_1 by Algorithm 2 from transition system ATS_1^3 and all the optimal alignments by Algorithm 3. The optimal alignments are corresponding to the paths from the initial node to the final nodes in ATS_1^3, as shown in Table 6.

No matter what activities the new trace contains and how the relations between them are, it can be decomposed into one of the four cases mentioned above. Then, we can deal with the alignment transition system according to the relation between the new trace and the log-based relation matrix. Eventually, the new system includes the alignments between the new trace and the model.

When a new trace not in the original event log is aligned with the process model, our approach does not need to generate a new search space completely but only needs to expand the existing search space appropriately. Hence, our approach has better scalability and applicability.

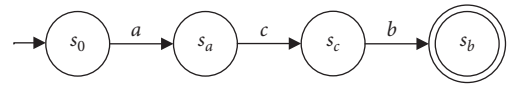

Figure 11: Transition system TS_8 of trace σ_8.

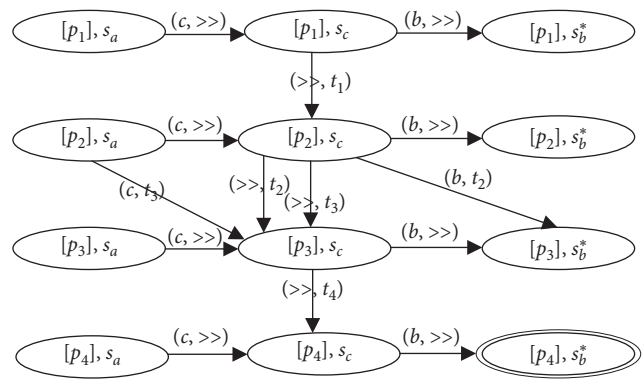

Figure 12: The newly added transitions in ATS_1^3 compared with ATS_1.

7. Case Studies

The greatest advantage of our approach is to deal with all traces in the event log at a time. It is more efficient than the approaches that can only deal with one trace every time. We integrate the following relations between events via the log-based relation matrix, so we can obtain all the alignments between batch traces and the model by an alignment transition system. In order to express the superiority of our approach, we compare it with *Alignment-One* approach. *Alignment-One* approach considers that the event log just has a trace and computes the alignments between a trace and the model via Algorithms 1–3. Taking a relatively complex process model for an example, this section provides some analysis results of our promoted approach. Compared with the results of *Alignment-One* approach, the superiority of our approach can be illustrated.

Both *Alignment-One* approach and our approach can be divided two steps: one is the generation of the search space; the other is the search for the optimal alignments. The scale of the search space determines the complexity of the search work to a certain extent. So, the complexity of the search space is critical for the performance of the alignment approaches. In this paper, it is deeply studied that the number and the size of the search spaces are generated during the execution of the alignment approaches, which indirectly illustrate their complexity. Hence, the case studies focus on the number and size of the search spaces. Through the research, the superiority of our approach is verified.

Table 6: Optimal alignments between trace σ_8 and model N_1.

Name	Path in ATS_1^3	Value
γ_{81}	$<(([p_1], s_0), (a, t_1), ([p_2], s_a)), (([p_2], s_a), (c, t_3), ([p_3], s_c)), (([p_3], s_c), (>>, t_4), ([p_4], s_c)), (([p_4], s_c), (b, >>), ([p_4], s_b^*))>$	$<(a, t_1), (c, t_3), (>>, t_4), (b,>>)>$
γ_{82}	$<(([p_1], s_0), (a, t_1), ([p_2], s_a)), (([p_2], s_a), (c, t_3), ([p_3], s_c)), (([p_3], s_c), (b, >>), ([p_3], s_b^*)), (([p_3], s_b^*), (\gg, t_4), ([p_4], s_b^*))>$	$<(a, t_1), (c, t_3), (b,>>), (>>, t_4)>$
γ_{83}	$<(([p_1], s_0), (a, t_1), ([p_2], s_a)), (([p_2], s_a), (c, >>), ([p_2], s_c)), (([p_2], s_c), (b, t_2), ([p_3], s_b^*)), (([p_3], s_b^*), (\gg, t_4), ([p_4], s_b^*))>$	$<(a, t_1), (c,>>), (b, t_3), (>>, t_4)>$

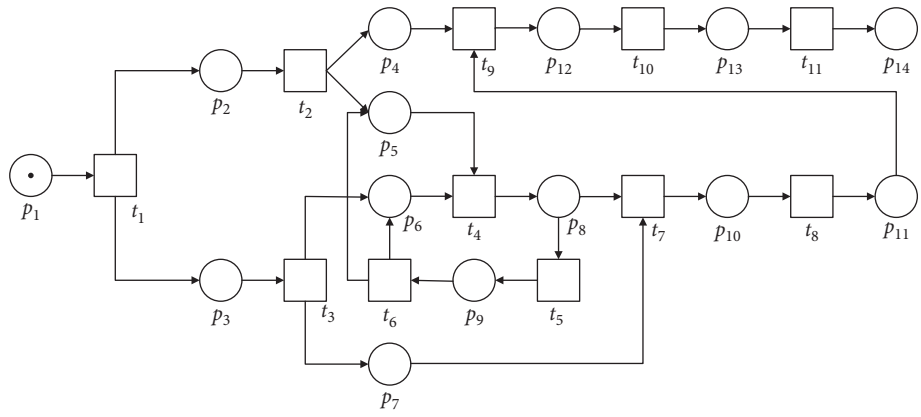

Figure 13: Petri net model N_S of the distributed control system for the inclined shaft of the coal mine.

In order to enhance the safety of the transportation of the coal mine, a distributed control system for the inclined shaft of the coal mine is introduced [32]. Petri nets are adopted to build the model for this system, as shown in Figure 13.

The model has a place named as p_1 that represents the start of the workflow and a place named as p_{14} that represents the end. Any place or transition in the model is on a path from p_1 to p_{14}. Moreover, the model has the properties, such as security, option to complete, proper completion, and no dead transitions. Hence, the model is considered to be a sound Petri net.

The actual meanings of places and transitions in Figure 13 are shown in Tables 7 and 8, respectively. In addition, each transition is mapped to an activity, as shown in Table 8. The name of each activity is a character, which is suitable for the representation of the control flow, i.e., trace.

Each event log is composed of some activity sequences generated randomly from the process model, and then we make several man-made noises for these sequences. The event logs are shown in Table 9.

The process model generates completely fit traces with different lengths, each of which contains about 6 to 15 activities. Noises are created by randomly deleting the activities from or adding them to the traces. However, when adding an activity, no activity beyond the given activity set appears. In this example, each activity in the traces is an element of the set $A = \{a, b, c, d, e, f, g, h, i, j, k\}$. Then, based on the standard likelihood cost function, the optimal alignments between all traces and the model are calculated. The number and the size of the search space are counted to compare *Alignment-One* approach with our approach.

Table 7: Places and their meanings of model N_S.

Place	Meaning
p_1	Beginning of task
p_2	Winch
p_3	Monitor system
p_4	End of dot
p_5	Dot signal
p_6	System idle
p_7	System normal
p_8	Receive of dot information
p_9	Malfunction
p_{10}	Successful road construction
p_{11}	End of answer
p_{12}	Ready
p_{13}	End of transportation
p_{14}	End of task

Table 8: Transitions and their meanings of model N_S.

Transition	Label	Meaning
t_1	a	Create and start the task
t_2	b	Dot
t_3	c	Detect the monitor system
t_4	d	Ask for the route
t_5	e	Diagnose the error information
t_6	f	Report error and return
t_7	g	Establish the route
t_8	h	Response the dot
t_9	i	Prepare for winch
t_{10}	j	Transport
t_{11}	k	Close the route

Table 9: Event logs.

Event log	Number of traces	Average length of traces	Traces
L_{s1}	5	6	[<a, b, c, d, g, h>, <a, b, c, d, g>, <a, b, c, d, g, h, i>, <a, d, c, g, i>, <a, b, d, c, g, h, i>]
L_{s2}	5	9	[<a, b, d, g, h, i, j, k>, <a, b, c, d, g, e, h, i, j, k>, <a, b, c, d, g, h, i, j, k>, <a, b, c, d, e, g, h, i, j, k>, <a, b, c, e, g, h, i, j>]
L_{s3}	5	12	[<a, b, c, d, e, f, d, g, h, i, j, k>, <a, b, c, d, e, f, g, h, i, j, k>, <a, b, c, d, d, e, f, d, g, h, i, j, k>, <a, b, c, d, e, f, d, e, g, h, i, j, k>, <a, b, c, d, f, e, d, g, h, i, j>]
L_{s4}	5	15	[<a, b, c, d, e, f, d, e, f, d, g, h, i, j, k>, <a, b, c, d, e, f, d, e, f, d, e, g, h, i, j, k>, <a, b, c, d, e, f, d, e, f, d, g, h, j, i>, <a, b, c, d, e, f, d, e, f, g, h, i, j, k>, <a, b, c, d, e, f, d, e, f, d, g, h, i, j, j, k>]

This instance contains four event logs, and each one contains five different traces. The average lengths of five traces in the four event logs are 6, 9, 12, and 15, respectively. The comparison results between *Alignment-One* approach and our approach are shown in Table 10.

According to Table 10, when an event log has five different traces, five alignment transition systems need to be established to get the optimal alignments between the traces in the log and the process model. However, the proposed approach in this paper needs only to establish an alignment transition system.

In *Alignment-One* approach, the number of states of the alignment transition system is related to not only the average length of the trace but also the number of the reachable states of the model. There is only one model in this instance, so the number of the reachable states is unchanged. When the average lengths of the traces in the log increase, the number of states of alignment transition systems obtained by *Alignment-One* approach increase linearly, which are proportional to the average lengths of the traces.

In our approach, the number of states of an alignment transition system is determined by that of different activities in the event log and that of reachable states of the process model. When the average lengths of the traces in the log are short, the number of states of the alignment transition systems in our approach is greater than that of *Alignment-One* approach. However, when the average lengths of the traces are greater than $|A|$ (A is the set of activities, and $|A|$ is the length of A.), the number of states of the alignment transition systems in our approach is a constant and its value is much less than that of *Alignment-One* approach. As shown in Table 10, when the average lengths of the traces are 12 and 15, the number of states of the alignment transition systems in our approach is fixed at 144. Even if the average lengths of the traces continue to increase, the number of states will remain 144 as long as each activity in the traces is the member of set A.

According to the abovementioned analysis, when there are m different traces in the event log, our approach only needs to compute an alignment transition system, while *Alignment-One* approach needs to compute m ones. Generally, our approach yields much fewer states than *Alignment-One* approach.

The comparison results show that our approach generates much smaller search spaces than *Alignment-One* approach. Hence, our approach outperforms *Alignment-One* approach.

Table 10: Comparison of alignment transition systems between *alignment-one* approach and our approach.

Event log	Number of transition systems		Average number of states	
	Alignment-one approach	Our approach	*Alignment-one* approach	Our approach
L_{s1}	5	1	84	96
L_{s2}	5	1	120	132
L_{s3}	5	1	156	144
L_{s4}	5	1	192	144

8. Conclusions

As more and more event logs are recorded in enterprise organizations, conformance checking between event logs and process models plays an increasingly important role in process mining. At present, as a significant technique of conformance checking, alignment is widely used in process discovery, precision checking, and process enhancement. Alignment can accurately locate the deviations and measure the fitness between the observed and modeled behaviors. Most of the existing alignment approaches can only compute an optimal alignment between the trace and the model, or even just a suboptimal alignment.

In order to solve the problems that the state space only includes the alignments between one trace and the model, this paper proposes a business alignment approach based on the transition system between relation matrices and Petri nets. This approach can generate an alignment transition system, which includes the alignments between all traces in the log and the model. No matter how many traces are involved in the log, this approach only needs to generate one alignment transition system. According to the search results of prefix alignments in the system, two algorithms are proposed to find an optimal alignment and all optimal alignments between all traces in the log and the model based on the given cost function, respectively.

The proposed approach in this paper effectively solves problems such as low efficiency and high memory occupancy. It improves the efficiency of calculating optimal alignments. The alignments between all traces in the log and the model are embodied in the alignment transition system. Our approach simplifies the search space which includes all the optimal alignments. In the future work, we will further study the relations between activities in the log so that the more reasonable relation matrices will be established. On

these grounds, the more efficient alignment approaches can be proposed.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by the Natural Science Foundation of China under Grant 61973180, Taishan Scholar Construction Project of Shandong Province, Key Research and Development Program of Shandong Province under Grant 2018GGX101011, and Natural Science Foundation of Shandong Province under Grants ZR2018MF001 and ZR2019MF033.

References

[1] W. M. P. van der Aalst, "Business process management: a comprehensive survey," *ISRN Software Engineering*, vol. 2013, Article ID 507984, 37 pages, 2013.

[2] W. M. P. van der Aalst, A. Adriansyah, and A. K. A. D. Medeiros, "Process mining manifesto," in *Lecture Notes In Business Information Processing*, pp. 169–194, Springer, Berlin, Germany, 2012.

[3] C. Li, M. Reichert, and A. Wombacher, "Mining business process variants: challenges, scenarios, algorithms," *Data & Knowledge Engineering*, vol. 70, no. 5, pp. 409–434, 2011.

[4] W. M. P. van der Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business Processes*, Springer, Berlin, Germany, 2011.

[5] P. Weber, B. Bordbar, and P. Tino, "A framework for the analysis of process mining algorithms," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 2, pp. 303–317, 2013.

[6] W. M. P. van der Aalst and C. Stahl, *Modeling Business Processes: A Petri Net Oriented Approach*, MIT Press, Cambridge, MA, USA, 2011.

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

[8] R. P. J. C. Bose and W. M. P. van der Aalst, "Process diagnostics using trace alignment: opportunities, issues, and challenges," *Information System*, vol. 37, no. 2, pp. 117–141, 2012.

[9] 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.

[10] Y. Y. Wang and Y. Y. Du, "Conformance checking based on extended footprint matrix," *Journal of Shandong University of Science and Technology(Natural Science)*, vol. 37, no. 2, pp. 9–15, 2018, in Chinese.

[11] A. Adriansyah, B. F. V. Dongen, and W. M. P. van der Aalst, "Towards robust conformance checking," in *Business Process Management Workshops*, pp. 122–133, Springer, Berlin, Germany, 2010.

[12] M. de Leoni, F. M. Maggi, and W. M. P. van der Aalst, "Aligning event logs and declarative process models for conformance checking," in *Proceedings of the International Conference on Business Process Management*, pp. 82–97, Springer, Tallinn, Estonia, September 2012.

[13] A. Adriansyah, *Aligning Observed and Modeled Behavior*, Eindhoven University of Technology, Eindhoven, Netherlands, 2014.

[14] M. L. van Eck, *Alignment-based Process Model Repair and its Application to the Evolutionary Tree Miner*, Eindhoven University of Technology, Eindhoven, Netherlands, 2013.

[15] D. Fahland and W. M. P. van der Aalst, "Model repair-aligning process models to reality," *Information Systems*, vol. 47, no. 1, pp. 220–243, 2015.

[16] A. Adriansyah, J. Munoz Gama, J. Carmona, B. F. V. Dongen, and W. M. P. van der Aalst, "Alignment based precision checking," in *Business Process Management Workshops*, pp. 137–149, Springer, Berlin, Germany, 2013.

[17] M. D. 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," *Information Systems*, vol. 47, no. 3, pp. 258–277, 2015.

[18] M. L. van Eck, J. C. A. M. Buijs, and B. F. van Dongen, "Genetic process mining: alignment-based process model mutation," in *Proceedings of the International Conference on Business Process Management*, pp. 291–303, Springer International Publishing, Haifa, Israel, September 2014.

[19] J. E. Cook and A. L. Wolf, "Software process validation," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 8, no. 2, pp. 147–176, 1999.

[20] X. Lu, D. Fahland, and W. M. P. van der Aalst, "Conformance checking in healthcare based on partially ordered event data," in *Emerging Technology and Factory Automation(ETFA)*, IEEE Press, Piscataway, NJ, USA, 2014.

[21] X. Lu, D. Fahland, and W. M. P. van der Aalst, "Conformance checking based on partially ordered event data," in *Proceedings of the International Conference on Business Process Management*, pp. 75–88, Barcelona, Spain, 2014.

[22] L. Wang, Y. Du, and W. Liu, "Aligning observed and modelled behaviour based on workflow decomposition," *Enterprise Information Systems*, vol. 11, no. 8, pp. 1207–1227, 2017.

[23] W. Song, X. Xia, H.-A. Jacobsen, P. Zhang, and H. Hu, "Efficient alignment between event logs and process models," *IEEE Transactions on Services Computing*, vol. 10, no. 1, pp. 136–149, 2017.

[24] Y. Tian, Y. Du, M. Li, D. Han, and Q. Hu, "Reduced alignment based on Petri nets," *Concurrency and Computation: Practice and Experience*, vol. 30, no. 23, Article ID e4411, 2018.

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

[26] Q. Hu, M. Liu, Z. Zhao, and J. Du, "A path detecting method to analyze the interactive compatibility of service processes based on WS-BPEL," *Concurrency and Computation: Practice and Experience*, vol. 30, no. 19, p. e4699, 2018.

[27] N. Ran, H. Su, and S. Wang, "An improved approach to test diagnosability of bounded Petri nets," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 2, pp. 297–303, 2017.

[28] N. Q. Wu and M. C. Zhou, "Modeling, analysis and control of dual-arm cluster tools with residency time constraint and activity time variation based on petri nets," *IEEE Transactions on Automation Science and Engineering*, vol. 9, no. 2, pp. 446–454, 2012.

[29] H.-C. Liu, X. Luan, Z. Li, and J. Wu, "Linguistic Petri nets based on cloud model theory for knowledge representation and reasoning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 4, pp. 717–728, 2018.

[30] G. Liu, "Some complexity results for the soundness problem of workflow nets," *IEEE Transactions on Services Computing*, vol. 7, no. 2, pp. 322–328, 2014.

[31] L. J. Wen, *Studies on Algorithms for Process Mining Based on WF-Net*, Tsinghua University, Beijing, China, 2007.

[32] W. W. Zhang and W. D. Liu, "Research of inclined shaft monitoring and control system of coal mine based on Petri net," *Computer Engineering and Applications*, vol. 48, no. 20, pp. 240–243, 2012, in Chinese.