Computing and Informatics, Vol. 45, 2026, 1017–1044, doi: 10.31577/cai_2026_4_1017

PETRI NET-BASED DEVIATION DETECTION BETWEEN EVENT LOGS AND PROCESS MODELS WITH MILESTONES

Yinhua Tian*, Ruizhe Zhang, Wen Liu

College of Intelligent Equipment
Shandong University of Science and Technology
Tai'an, 271000, China
e-mail: {skd994121, 202383230026, 202483230017}@sdust.edu.cn

Yuyue Du

College of Computer Science and Engineering
Shandong University of Science and Technology
Qingdao, 266000, China
e-mail: yuyuedu@sdust.edu.cn

Abstract. The designation of milestone activities in business processes can restrict the effective occurrence of cases in logs and remove noise reasonably. A deviation detection method is presented between Petri nets and logs using milestone activities to improve computational efficiency. An original model is constructed through some operations between two Petri nets, where one is the process model and the other is the log model created by mapping the activities in the case to the transitions. First, the transitions labeled with the milestones compose the synchronous transitions by synchronous composition. Next, other transitions with the same labels produce the log, model and synchronous transitions by product. Then, the transitions labeled with the activities that are merely observed in the model directly map to the transitions. Finally, the transitions labeled with the activities which are modeled only in the Petri net directly map to the model transitions, and the transitions labeled with no activities directly map to the invisible transitions. The new model is the search space for deviation detection. An algorithm is presented to detect the deviations between the Petri net based on the given cost and the case. The experimental

* Corresponding author

results show that the more milestone activities in the business process, the higher computational efficiency of deviation detection.

Keywords: Petri nets, milestone activities, event logs, process models, deviation detection

Mathematics Subject Classification 2010: 68Q85, 93A30

1 INTRODUCTION

In the past few years, due to the increased accessibility of event data, the field of process mining has undergone rapid development [1, 2]. Process mining is a key technology to discover Petri nets from logs, and it has become increasingly important in the field of business process management (BPM) [3, 4]. Many companies have incorporated process mining into their product suites. Research in process mining is of significant importance for implementing new business processes and analyzing and improving existing processes. Hence, process mining is a worldwide current trending research area in relevant disciplines.

Process mining aims to learn useful information extracted from logs, and then identify, track, and enhance real-world operational workflows [5]. Here, event logs refer to the data of event execution accumulated and collected by the BPM systems during their implementation, which usually contain the actual execution information of the processes, for instance the names of activity, timestamps, executors, etc. Process mining's primary application is to reveal the control flow structure of operational workflows, next by analyzing the activities recorded in the event logs, a Petri net that can depict the causal dependencies between the activities is automatically built [6, 7, 8, 9, 10].

To evaluate the models' quality, it is essential to check the compliance between Petri nets and logs [11]. The prerequisite for conformance checking is to detect the anomalies between the case in the logs and the Petri net using a given cost function [12]. The occurrence of deviations may have a negative impact on the efficiency, quality, and profits of the enterprise. Therefore, conformance checking of business processes has become crucial for enterprises [13]. By establishing effective business process deviation detection methods, it is possible to monitor non-compliant situations in real time during enterprise operations, helping enterprises to identify problems in a timely manner and resolve potential crises, thereby reducing losses, improving customer satisfaction, and enhancing competitiveness. The process of deviation detection between traces and Petri nets is essentially an optimization problem [14]. At present, the most classic deviation detection method is A* alignment algorithm proposed by Adriansyah et al. [15]. When detecting the deviations, the fitness is used to quantify the aligned and deviated activities in the traces and Petri nets. If the Petri net can completely align every trace in the log, i.e., meaning

no deviations exist between the two. Then the log can be replayed on the Petri net, and the fit degree is 1.

To determine the effective traces in the log and remove noise from the log, the concept of milestone is proposed [16]. Milestone activities refer to the activities that must be observed in the case and synchronously modeled by the Petri net. The trace that does not contain milestone activities is regarded as noise, and milestone activities are not allowed to be skipped during the process model running. Hence, milestone activities are essential in both the case and the Petri net, and should be synchronized. The introduction of milestone activities is the minimum requirement to avoid the process model selecting a meaningless transition firing sequence in the process of deviation detection. Based on the concept of milestone activities, this paper proposes a method with a general purpose but lower complexity, which is called deviation detection using milestones (DDuM for short). The execution procedure of this method is shown in Figure 1.

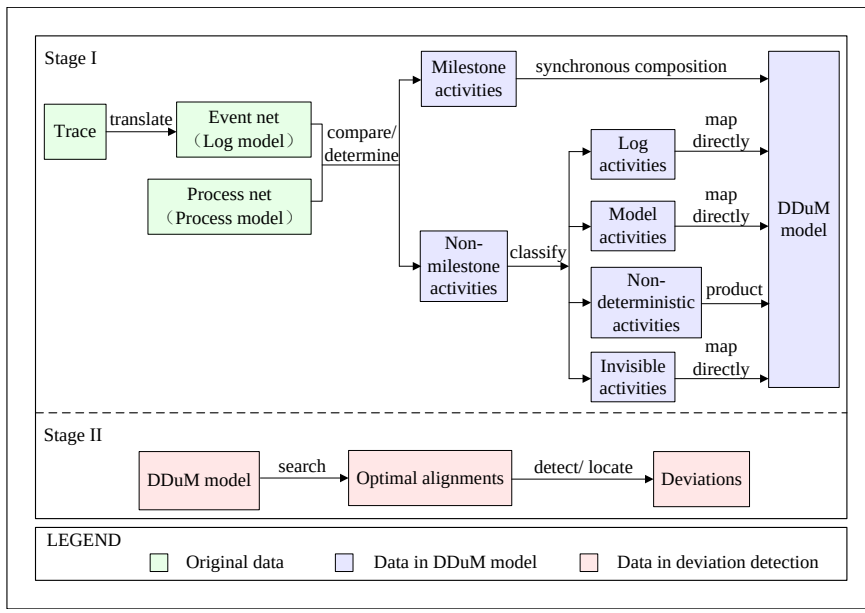


Figure 1. The execution procedure of DDuM method

The whole execution procedure of the DDuM method is possible to divide it into two stages. In stage I, firstly, the activities in the given trace are translated into the transitions to construct an event net with sequential structures. Secondly, according to the milestone activities between the process net that models the business process and the event net, synchronous composition of the transitions with the same milestones is performed. Then, the rest of the activities come in four varieties: the log activities, which are not modeled in the model but detected in the case; the

invisible activities or model, which are not detected in the case but modeled in the model; and the non-deterministic activities, which are observed in the trace and modeled in the model. The transitions with the log, model and invisible activities perform the direct mapping operations, while the product of the transitions with the non-deterministic activities is executed. Finally, a new Petri net named DDuM model is constructed. In stage II, considering the DDuM model as the search object, a standard likelihood cost function is used to find the shortest path, which corresponds to the optimal alignment. The deviations between the original case and the Petri net can be detected and located through the optimal alignment.

This paper's proposed DDuM method leverages the advantage of milestones mentioned in the MTCG (Milestone Transitive Closure Graph) method [16]. The overall design idea is an improvement of A* alignment algorithm. Due to the introduction of milestone activities, product between some synchronous transitions in A* alignment algorithm is simplified to synchronous composition, which reduces the size of the search space and improves the efficiency to calculate the optimal alignment. Through the case studies, it can be demonstrated that the more milestone activities in the business process, the search space is smaller and the effectiveness of the algorithm is higher.

2 RELATED WORK AND BASIC KNOWLEDGE

Deviation detection is a key area of research within business process management. In recent years, scholars have conducted extensive research on this topic, and business process deviation detection methods can be roughly divided into process model-based methods and machine learning based methods.

With the development of machine learning, business process deviation detection methods based on machine learning have been widely studied. Pauwels et al. extend the Bayesian model using existing techniques, which do not require any prior knowledge about data and attributes and can identify the root cause of anomalies [17]. Vertuam et al. use Word2Vec for feature representation and propose an autonomous and evolutionary online clustering algorithm for compliance checking [18]. Nguyen et al. compare the accuracy of decision trees, K-nearest neighbors, and neural networks in compliance checks, and the results show that neural networks have better performance [19]. Vijayakamal et al. propose a compliance inspection method based on deep learning encoders, which improves the ability of compliance inspection compared to machine learning methods [20]. Lahann et al. use LSTM to detect anomalies and make improvements in data processing, network architecture, and anomaly scoring, improving the quality of detection [21]. Elaziz et al. employ an LSTM recurrent neural network integrated with a self-attention mechanism to address the long-term dependency issues in business process events. They enhance the learning model using a Variational Autoencoder (VAE) [22]. Tian et al. propose a conformance checking method of business processes, which is based on improved BERT and lightweight CNN [23].

Although the flourishing development of machine learning methods has made the research on bias detection methods based on machine learning increasingly popular. However, it has the following problems:

1. Machine learning requires a large amount of training time, especially deep learning that require high-performance computing devices such as servers;
2. Machine learning-based methods require large-scale datasets, and the larger the sample size, the better the learning effect. However, medium-scale event logs are more likely to cause overfitting;
3. Machine learning-based methods lack interpretability and cannot theoretically demonstrate the correctness of detection results;
4. Machine learning-based methods are detached from process models and rely solely on event logs for bias detection, resulting in detection results that tend to define outliers as biases, which is unreasonable.

Given this, although process model-based bias detection methods started early, they are still a current research hotspot. In many studies, the deviation detection results based on process models are used as a standard to measure the accuracy of other methods. Hence, the methods of deviation detection based on process models are very practical and widely used. Since the related method named alignment was proposed, a lot of alignment-based research has been carried out. Cook et al. present a method for comparing traces to Petri net, which allows for a quantitative assessment of their similarity [24]. Adriansyah et al. provide a methodology to align the modeled and observed behaviors [15]. Lu et al. propose a partially ordered event data-based approach for conformance checking [25]. Feng et al. present a new deviation detection approach between logs and Petri net using min-cost transition systems [26]. Bloemen et al. present a method to align observed and modeled behaviors by maximizing synchronous movements and implementing milestones, which is called the MTCG method [16].

Meanwhile, there is more literature on deviation detection in process mining. By analyzing the existing methods, it is found that the existing algorithms are either suitable for a specific purpose or prohibitively complex. Hence, a new method is presented to improve the efficiency of deviation detection. Furthermore, the paper will use several definitions, mainly including multisets, traces, Petri nets, alignments, optimal alignments, and so on. Please refer to references [2, 6, 27, 28, 29, 30] for details; they will not be elaborated further herein.

3 DDUM METHOD

In this section, to demonstrate how the DDuM model is created, a provided trace and a process model are used as examples, so as to express the idea of the method more clearly and vividly.

3.1 Process Model

According to the file operation in the computer system, a Petri net model is built, as seen in Figure 2 [16]. Here, activity a means to open a file, b to parse a file, c to increment the file counter by 1, d to close a file, and e to indicate that no file was found.

Based on the given process model, some specific activities are designated as milestones. For example, we specify activities a and b as milestones in model N_1. The set of the milestones in this process means that it is not allowed to find any required file in the system and the reasonable observed behavior must include two operations: opening the file and parsing the file. Because the order in which the milestones occur is fixed, the relation matrix is used to represent the milestones and their relations. The milestone relation matrix not only records all the milestones, but also indicates their precedence relations.

Definition 1. The milestone relation matrix is a matrix with the labels $S \times S$ in its row and column, denoted by $M : S \times S \to \{\text{“\#”}, \text{“} \to \text{”}\}$. The elements in the matrix are as follows:

1. For $\forall\alpha(t_i) \in S$ and $\forall\alpha(t_j) \in S$, if $\exists t'_1 t'_2 \dots t_i \dots t_j \dots t'_m \to m_{\text{i}} \xrightarrow{t'_1 t'_2 \dots t_i \dots t_j \dots t'_m} m_{\text{f}}$, $M[\alpha(t_i)][\alpha(t_j)] = \text{“} \to \text{”}$, denoted by $\alpha(t_i) \to \alpha(t_j)$, where $1 \le i$ and $j \le n$;
2. For $\forall\alpha(t_i) \in S$ and $\forall\alpha(t_j) \in S$, if $\nexists t'_1 t'_2 \dots t_i \dots t_j \dots t'_m \to m_{\text{i}} \xrightarrow{t'_1 t'_2 \dots t_i \dots t_j \dots t'_m} m_{\text{f}}$, $M[\alpha(t_i)][\alpha(t_j)] = \text{“\#”}$, denoted by $\alpha(t_i)\#\alpha(t_j)$, where $1 \le i$ and $j \le n$.

Definition 1 displays a reasonable order relation between the milestones. If an activity can occur before the other when running the process model, it has a casual relation with the following one. However, if an activity cannot occur in front of the other, it has no relation with that one. In this example, the milestone activities include a and b, which can form the set $S_1 = \{a, b\}$. The interactions of a and b can be represented as the milestone relation matrix M_1, as shown in (1).

$$\begin{array}{c} \\ a \\ b \end{array} \begin{array}{c} \begin{array}{cc} a & b \end{array} \\ \begin{bmatrix} \# & \to \\ \# & \# \end{bmatrix} \end{array}. \tag{1}$$

In model N_1, a whole transition firing sequence $\langle t_1, t_2, t_3, t_4 \rangle$ exists, which can be mapped to the activity sequence $\langle a, b, c, d \rangle$. $S_1 = \{a, b\} \subseteq \partial_{\text{set}}(\langle a, b, c, d \rangle)$ is workable, and the relation between a and b in (1) conforms to that in $\langle a, b, c, d \rangle$. So, M_1 is a reasonable milestone relation matrix.

Definition 2. A milestone system is denoted by $MS = (S, M)$, where S is a milestone set on A and M is a milestone relation matrix on S.

The milestone system includes the set and relation matrix. In fact, the milestone relation matrix not only records all the milestones, but also indicates their precedence relations.

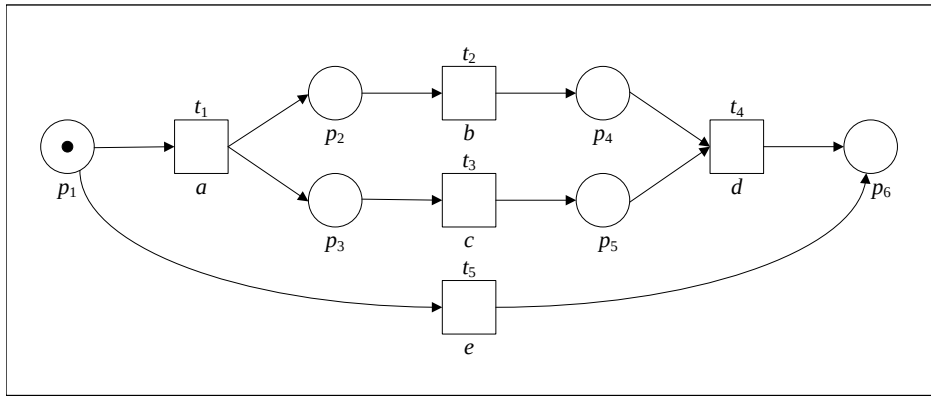


Figure 2. Process model N_1

3.2 Log Model

Given trace $\sigma_1 = \langle a, c, b, e, f \rangle$, it contains activities a and b. Although a is not adjacent to b, b follows a and a precedes b. The relation between a and b conforms to the constraints of the milestone relation matrix in model N_{p}. Hence, the trace is suitable to be aligned with the business process using the milestone relation matrix. This alignment condition that the trace must meet is called the DDuM criterion.

Definition 3. M is a milestone relation matrix on S. The trace must meet the alignment criterion for the business process with milestone system. This criterion is called as DDuM criterion, including:

1. For $\forall \alpha(t) \in S$, $\alpha(t) \in \partial_{\mathrm{set}}(\sigma)$;
2. For $\forall \alpha(t_p) \in S$ and $\forall \alpha(t_q) \in S$, suppose that $\alpha(t_p) = \sigma[i] \wedge \alpha(t_q) = \sigma[j]$, if $i < j$, $M[\alpha(t_p)][\alpha(t_q)] = \text{“} \rightarrow \text{”}$.

The DDuM criterion is the basis to investigate whether case in log can be aligned with the Petri net by the DDuM method, which is a formal expression of the alignment condition. When the given trace satisfies the DDuM criterion, it can be aligned by DDuM method; otherwise, it is considered as the noise and will be removed from the log. In this example, $\sigma_2 = \langle a, b, c, d \rangle$, $S_1 = \{a, b\}$, $a = \sigma_2[1] \wedge b = \sigma_2[2]$, $1 < 2$. And in (1), $M[a][b] = \text{“} \rightarrow \text{”}$. Hence, trace σ_2 meets the DDuM criterion and can be aligned using the DDuM method. Given that $\sigma_3 = \langle c, b, a, e, f \rangle$, $b = \sigma_3[2] \wedge a = \sigma_3[3]$ and $2 < 3$, but $M[a][b] =$ “#” in (1), so trace σ_3 does not meet the DDuM criterion and cannot be aligned using the DDuM method.

Next, all the activities in trace σ_1 are mapped to the transitions one by one. A place is added as the initial place, from which an arc is inserted to the transition that corresponds to the first activity in the case; a place is added as the final place, to which an arc is inserted from the transition corresponding to the last activity in

the trace; a place is added between the transitions corresponding to any two adjacent activities, from which an arc is inserted to the transition that corresponds to the prior activity and to which an arc is inserted from the transition that corresponds to the posterior activity. After the above operations, log model N_2 corresponding to trace σ_1 is obtained, as shown in Figure 3.

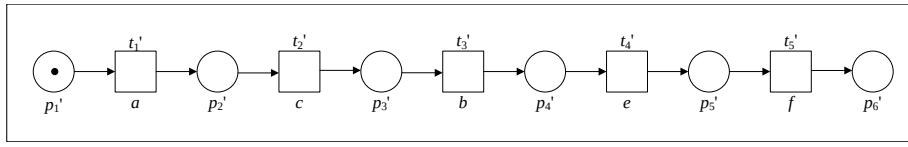


Figure 3. Log model N_2

3.3 DDuM Model

Comparing the activities between the Petri net and the log model according to the milestone set, it is found that the activities between the two models come in five varieties: milestone activities, log activities, model activities, invisible activities and non-deterministic activities. Among them, log, model, invisible and non-deterministic activities are all non-milestone ones. Obviously, all milestone activities can be mapped to synchronous moves. Except for milestone activities, non-deterministic activities occur in both the log model and the Petri net, but they probably cannot synchronize in actual alignment. Invisible activities denoted by "τ" are unique to the Petri net, which cannot take place in the log model, so their operation is the same as that of the model activities.

For different activities, different operations should be carried out when constructing the DDuM model. Using the provided Petri net and log model as an illustration, the generation process of the DDuM between the two is illustrated according to different activity types.

Generally, the log model is represented by $N_{\mathrm{L}} = (P_{\mathrm{L}}, T_{\mathrm{L}}; F_{\mathrm{L}}, \alpha_{\mathrm{L}}, m_{\mathrm{iL}}, m_{\mathrm{fL}})$, the process model is $N_{\mathrm{P}} = (P_{\mathrm{P}}, T_{\mathrm{P}}; F_{\mathrm{P}}, \alpha_{\mathrm{P}}, m_{\mathrm{iP}}, m_{\mathrm{fP}})$, the DDuM model is $N_{\mathrm{D}} = (P_{\mathrm{D}}, T_{\mathrm{D}}; F_{\mathrm{D}}, \alpha_{\mathrm{D}}, m_{\mathrm{iD}}, m_{\mathrm{fD}})$, and the milestone set is S. According to the construction rules of the DDuM model, the place set in the DDuM model is the union of that of the log and Petri net; while the transition set and arc set are generated using the specific operations for different move types.

1. Milestone activities

   Milestone activities refer to the ones that must be observed in the trace and modeled synchronously by the process model, which must correspond to synchronous movement in the alignment. All activities in the given milestone set are milestones. When constructing the DDuM model, the transitions labeled with the milestones in the Petri net and trace are firstly determined, and then

the two transitions can perform the synchronous composition. The so-called synchronous composition is to combine two transitions into one. The union of the presets and postsets from the original transitions makes up the presets/postsets of the original transition. The synchronous composition of transitions with milestones is shown in Figure 4.

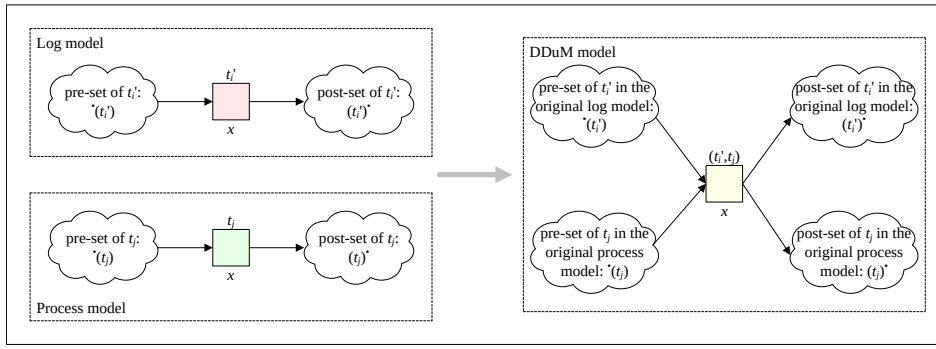


Figure 4. Synchronous composition of transitions with milestone activities

Activity a in the trace and process model is a milestone when it meets the following condition: $\forall a \in Y$. When constructing the DDuM model, the specific operations for the milestone activities are shown in Algorithm 1.

Algorithm 1 Synchronous composition of the milestone activities

Input: Milestone activity a;
Output: New transition T_{D} and its arcs α_{D}.

1: $T_{\mathrm{D}} \leftarrow T_{\mathrm{D}} \cup \{(t_i', t_j)\}$;
2: $\alpha_{\mathrm{D}}((t_i', t_j)) \leftarrow a$;
3: $\bullet(t_i', t_j) \leftarrow \bullet(t_i') \uplus \bullet(t_j)$;
4: $(t_i', t_j)\bullet \leftarrow (t_i')\bullet \uplus (t_j)\bullet$;
// to generate the new synchronous transition and determine its arcs

2. Log activities

Compared the non-milestone activities in the trace and process model, if an activity can only be observed in the trace and cannot be modeled in the process model, it is a log activity. Log activities can only be mapped to log moves in the alignment. When constructing the DDuM model, the transitions labeled with such activities are directly translated into the log transitions. Figure 5 displays the direct mapping of transitions with log activities.

Activity a in the case and Petri net is a log activity when it meets the following condition: $a \in \{\alpha(t_i') | t_i' \in T_{\mathrm{L}}(1 \leq i \leq |T_{\mathrm{L}}|)\} \wedge a \notin \{\alpha(t_j) | t_j \in T_{\mathrm{P}}(1 \leq j \leq |T_{\mathrm{P}}|)\}$.

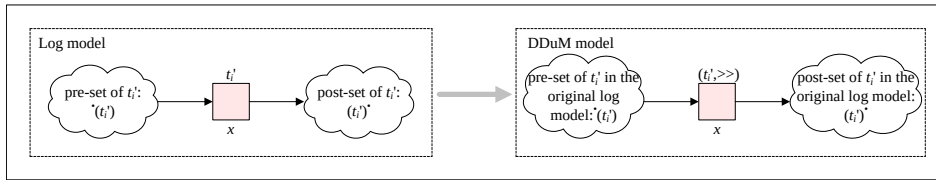


Figure 5. Direct mapping of transitions with log activities

When constructing the DDuM model, the specific operations for the log activities are shown in Algorithm 2.

Algorithm 2 Direct mapping of the log activities

Input: Log activity a;

Output: New transition T_{D} and its arcs α_{D}.

1: $T_{\mathrm{D}} \leftarrow T_{\mathrm{D}} \cup \{(t_i', >>)\}$;
2: $\alpha_{\mathrm{D}}((t_i', >>)) \leftarrow a$;
3: $\dot{}(t_i', >>) \leftarrow \dot{} (t_i')$;
4: $(t_i', >>)\dot{} \leftarrow (t_i')\dot{}$;
// to generate the new log transition and determine its arcs

3. Model or invisible activities

Comparing non-milestone activities in the trace and process model, if a process can be represented but an activity cannot be seen in the case, it is a model or invisible activity. Model activities can only be mapped to model moves in the alignment. The invisible activity refers to the activity of the invisible transition in the Petri net, denoted by "τ". Invisible activities can only be mapped to invisible moves in the alignment.

Although model activities perform the same operations with invisible activities when constructing the DDuM model, their actual meanings are quite different. Model activities are such behaviors that have activity names with some specific meanings and can only be modeled in the Petri net, but unfortunately can not be observed in the trace. However, invisible activities are such behaviors that can be modeled but have no activity names and are just labeled with "τ" in the process model. Even if invisible activities occur in the Petri net, nothing will be recorded. In the circumstance, it is reasonable and undoubted that they cannot be observed in the trace. Hence, model activities are unfit behaviors, while invisible activities are fit behaviors. Meanwhile, model moves means deviations between the trace and process model, but invisible moves do not.

When constructing the DDuM model, the transitions labeled with the model activities are directly translated into the model transitions. However, the transi-

tions labeled with the invisible activities are directly translated into the invisible transitions. Figure 6 displays the direct mapping of transitions with model or invisible activities.

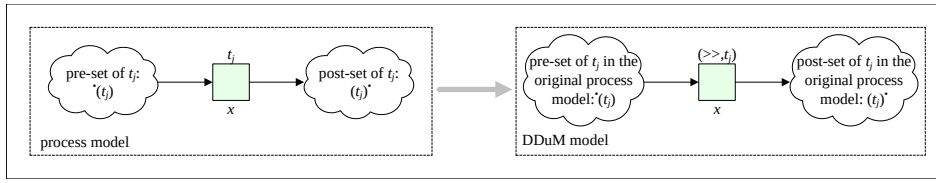


Figure 6. Direct mapping of transitions with model or invisible activities

Activity a in the trace and Petri net is a model activity when it meets the following condition: $a \notin \{\alpha(t_i')|t_i' \in T_{\mathrm{L}}(1 \leq i \leq |T_{\mathrm{L}}|)\} \wedge \alpha \in \{\alpha(t_j)|t_j \in T_{\mathrm{P}}(1 \leq j \leq |T_{\mathrm{P}}|)\}$. However, a is an invisible activity when it meets the following conditions: $a = \tau$. When constructing the DDuM model, the specific operations for the model or invisible activities are shown in Algorithm 3.

Algorithm 3 Direct mapping of the model or invisible activities

Input: Model or invisible activity a;
Output: New transition T_{D} and its arcs α_{D}.

1: $T_{\mathrm{D}} \leftarrow T_{\mathrm{D}} \cup \{(>>, t_j)\}$;
2: $\alpha_{\mathrm{D}}((>>, t_j)) \leftarrow a$;
3: $\cdot(>>, t_j) \leftarrow \cdot\ (t_j)$;
4: $(>>, t_j)^\cdot \leftarrow (t_j)^\cdot$;
// to generate the new model or invisible transition and determine its arcs

4. Non-deterministic activities

Compared the non-milestone activities in the trace and process model, if an activity can be observed in the trace as well as be modeled in the Petri net, it is a non-deterministic activity. Although non-deterministic activities can occur in both the case and the Petri net, they may not be synchronous. Hence, the non-deterministic activity may be mapped to the synchronous move or the log and model movement in the alignment. To ensure that all possible alignments are taken into account, the product of the log and process models should be performed based on the transitions with the non-deterministic activities. The product of transitions with non-deterministic activities is shown in Figure 7.

Activity a in the trace and process model is a non-deterministic activity when it meets the following condition: $a \in \{\alpha(t_i')|t_i' \in T_{\mathrm{L}}(1 \leq i \leq |T_{\mathrm{L}}|)\} \wedge a \in \{\alpha(t_j)|t_j \in T_{\mathrm{P}}(1 \leq j \leq |T_{\mathrm{P}}|)\} \wedge a \notin Y$. When constructing the DDuM model, the specific operations for non-deterministic activities are shown in Algorithm 4.

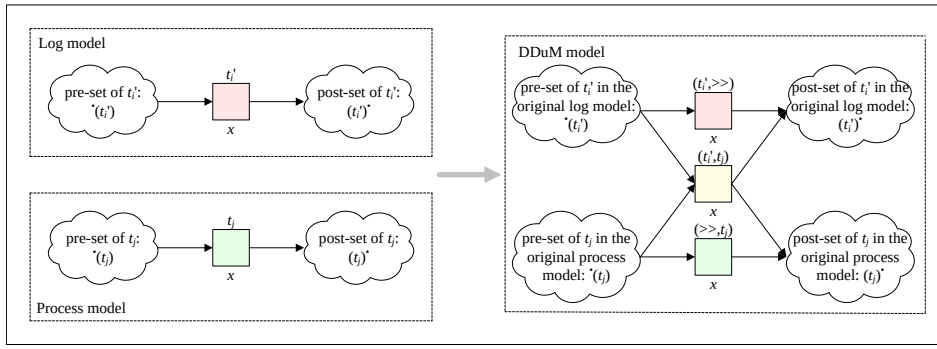


Figure 7. Product of transitions with non-deterministic activities

Algorithm 4 Product of the non-deterministic activities

Input: Non-deterministic activity a;

Output: New transition T_{D} and its arcs α_{D}.

1: **CALL** Algorithm 1; // in order to construct the new synchronous transition and determine its arcs
2: **CALL** Algorithm 2; // in order to construct the new log transition and determine its arcs
3: **CALL** Algorithm 3; // in order to construct the new model or invisible transition and determine its arcs

Compared the non-milestone activities in the trace and process model, if an activity can be observed in the trace as well as be modeled in the Petri net, it is a non-deterministic activity. Although non-deterministic activities can occur in both the trace and the Petri net, they may not be synchronous. Hence, the non-deterministic activity may be mapped to the synchronous move or the log and model movement in the alignment. To ensure that all possible alignments are taken into account, the product of the log and process models should be performed based on the transitions with the non-deterministic activities. The product of transitions with non-deterministic activities is shown in Figure 7.

5. Construction of DDuM model

After classifying the different activities in the trace and process model and analyzing the specific operations that different activities should be performed when generating the DDuM model, the whole process is shown in Algorithm 5 to build the DDuM model utilizing the provided log model, process model and milestone set.

This algorithm consists of two outermost loops in sequence. At first, the first loop is to deal with milestone activities. The execution time of this loop is mainly required to determine the transitions labeled with milestone activities in the log

Algorithm 5 Construction of DDuM model between the log model and the Petri net based on the milestone set

Input: Log model $N_{\mathrm{L}} = (P_{\mathrm{L}}, T_{\mathrm{L}}; F_{\mathrm{L}}, \alpha_{\mathrm{L}}, m_{\mathrm{iL}}, m_{\mathrm{fL}})$, process model N_{P}=$(P_{\mathrm{P}}, T_{\mathrm{P}}; F_{\mathrm{P}}, \alpha_{\mathrm{P}}, m_{\mathrm{iP}}, m_{\mathrm{fP}})$, and milestone set $S = \langle a_1, a_2, \ldots, a_{\mathrm{n}} \rangle$;
Output: DDuM model $N_{\mathrm{D}} = (P_{\mathrm{D}}, T_{\mathrm{D}}; F_{\mathrm{D}}, \alpha_{\mathrm{D}}, m_{\mathrm{iD}}, m_{\mathrm{fD}})$.

1: $P_{\mathrm{D}} \leftarrow P_{\mathrm{L}} \cup P_{\mathrm{P}}$; $T_{\mathrm{D}} \leftarrow \varnothing$; $F_{\mathrm{D}} \leftarrow \varnothing$; $m_{\mathrm{iD}} \leftarrow m_{\mathrm{iL}} \uplus m_{\mathrm{iP}}$; $m_{\mathrm{fD}} \leftarrow m_{\mathrm{fL}} \uplus m_{\mathrm{fP}}$; // to initialize
2: **for** $\forall a \in Y$ **do**
3: **for** $\forall t_i' \in T_{\mathrm{L}}$ **do**
4: **if** $(\alpha(t_i')=a)$ **then**
5: EXIT
6: **end if**
7: **end for** // to check all the transitions in the log model and find the transition with the activity labeled by a
8: **for** $\forall t_j \in T_{\mathrm{P}}$ **do**
9: **if** $\alpha(t_j) = a$ **then**
10: EXIT
11: **end if**
12: **end for** // to check all the transitions in the model model and find the transition with the activity labeled by a
13: **CALL** Algorithm 1; // the synchronous composition of the milestone activity
14: **end for**
15: **for** $\forall a \notin S$ **do**
16: **if** $a \in \{\alpha(t_i')|t_i' \in T_{\mathrm{L}}(1 \le i \le |T_{\mathrm{L}}|)\} \wedge \alpha \notin \{\alpha(t_j)|t_j \in T_{\mathrm{P}}(1 \le j \le |T_{\mathrm{P}}|)\}$ **then**
17: **CALL** Algorithm 2; // the direct mapping of the log activity
18: **end if**
19: **if** $(a \notin \{\alpha(t_i')|t_i' \in T_{\mathrm{L}}(1 \le i \le |T_{\mathrm{L}}|)\} \wedge a \in \{\alpha(t_j)|t_j \in T_{\mathrm{P}}(1 \le j \le |T_{\mathrm{P}}|)\})$ **OR** $(a = \tau)$ **then**
20: **CALL** Algorithm 3; // the direct mapping of the model or invisible activity
21: **end if**
22: **if** $a \in \{\alpha(t_i')|t_i' \in T_{\mathrm{L}}(1 \le i \le |T_{\mathrm{L}}|)\} \wedge \alpha \in \{\alpha(t_j)|t_j \in T_{\mathrm{P}}(1 \le j \le |T_{\mathrm{P}}|)\}$ **then**
23: **CALL** Algorithm 4; // the product of the non-deterministic activity
24: **end if**
25: **end for**
26: **return** $N_{\mathrm{D}} = (P_{\mathrm{D}}, T_{\mathrm{D}}; F_{\mathrm{D}}, \alpha_{\mathrm{D}}, m_{\mathrm{iD}}, m_{\mathrm{fD}})$;

and Petri net, whose complexity is $O(|T_{\mathrm{L}}| + |T_{\mathrm{P}}|)$, where $|T_{\mathrm{L}}|$ means that there are $|T_{\mathrm{L}}|$ transitions in the log model and $|T_{\mathrm{P}}|$ means that there are $|T_{\mathrm{P}}|$ transitions in the Petri net. Furthermore, the second loop deals with non-milestone activities. The execution time of this loop is mainly required to compare the similarities and discrepancies between the activities in the model and those in

the Petri net, whose complexity is $O((|T_L| - |S|)^*(|T_P| - |S|))$. Assuming that milestone activities are far less than transitions in the log and Petri net, which means $|Y| << |T_L|$ and $|Y| << |T_P|$, the second loop's time complexity is about $O(|T_L|^*|T_P|)$. As the size of the log model and Petri net increases, especially the transitions, $|T_L| + |T_P| << |T_L| * |T_P|$ holds. Hence, the time of Algorithm 5 is $O(|T_L|^*|T_P|)$.

Taking process model N_1, log model N_2 and milestone set S as examples, firstly, all the activities are studied and classified into various categories in order to determine and perform the related operations. Activities a and b in the milestone set are milestone activities; activity f in the log model is a log activity; activity d in the Petri net is a model activity; activities c and e in both the log model and the Petri net are non-deterministic activities. The DDuM model N_3 between process model N_1 and log model N_2 based on milestone set S_1 can be constructed using Algorithm 5, which is shown in Figure 8.

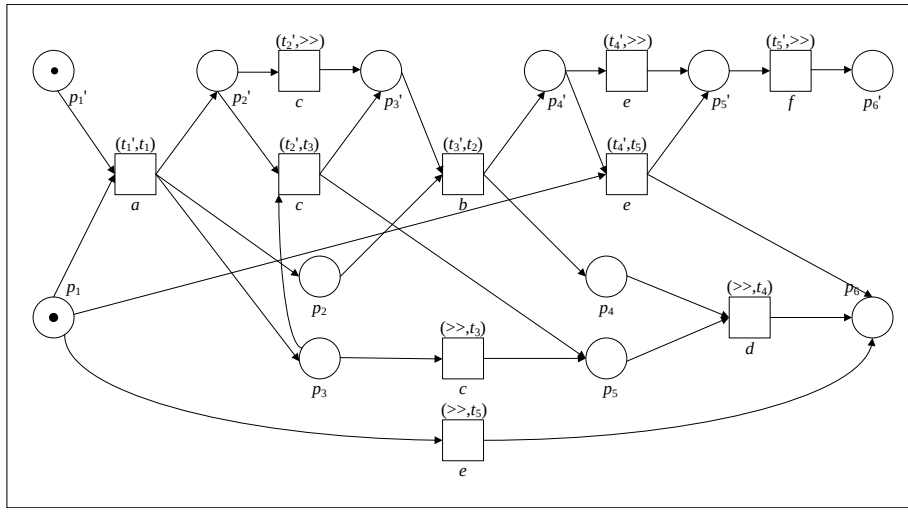


Figure 8. DDuM model N_3 between process model N_1 and log model N_2 based on milestone set S_1

3.4 Deviation Detection

Given the Petri net and the trace, the DDuM model between the log and the Petri net with the milestone set can be constructed according to Algorithm 5, where the log model is generated by mapping the activities in the trace to the transitions. Through the construction of the model, the observation of activities in the trace, the modeling of activities in the process model and all possible comparison results between the two can be embodied on the transitions of the DDuM model. The

transitions of the DDuM model come in four categories: synchronous transitions, represented in the form $(t_i^{'}, t_j)$; log transitions, represented in the form $(t_i', >>)$; model transitions and invisible transitions, represented in the form $(>>, t_j)$, where $1 \leq i \leq |T_{\mathrm{L}}|$ and $1 \leq j \leq |T_{\mathrm{P}}|$. By means of the mapping function $\alpha_{\mathrm{L}}(t_i')$ of the log model, the above-mentioned transitions can be mapped to moves in the alignment in turn: synchronous moves, denoted by $(\alpha_{\mathrm{L}}(t_i'), t_j)$; log moves, denoted by $(\alpha_{\mathrm{L}}(t_i'), >>)$; model moves and invisible moves, denoted by $(>>, t_j)$.

The standard likelihood cost function to assign advisable costs to the various moves in this paper: the cost of the log movement and model movement is 1, while that of the synchronous move and invisible move is 0. In DDuM model, there is a strict one-to-one correspondence between the transitions and the moves, so the similar cost function can be created to map the transitions to the related cost: the cost of the log transition and model transition is 1, while that of the synchronous transition and invisible transition is 0. Then, the DDuM model is further evolved into the model with the cost function.

In the DDuM model, after the transitions are associated with the cost, the cost is incurred when the transitions are fired, and the total cost of a complete transition firing sequence can be obtained by adding up the cost of all transitions in the firing sequence from the initial to the final marking. This total cost represents the cost of an alignment between the Petri net and the trace. The search of the optimal alignment between the trace and Petri net is translated into the problem of search for the min-cost transition firing sequence in the DDuM model. The effective solution is the state space search. The calculation of the min-cost transition firing sequence can be solved by calculating the shortest path. Based on the DDuM model, the specific process of search for the optimal alignment is shown in Algorithm 6.

The main goal of Algorithm 6 is to search for the transition firing sequence with the minimum cost in the weighted Petri net. This algorithm adopts the most intuitive search method. To ensure that the first complete transition firing sequence has the least cost, the breadth-first search strategy is used in the search process. The Petri net's complexity and Algorithm 6's temporal complexity are connected, especially its reachable markings.

Taking DDuM model N_3 as an example, Algorithm 6 is used to calculate an optimal alignment between trace σ_1 and Petri net N_1 with milestone set S_1 based on standard likelihood cost function $lc((a,t))$. Algorithm 6 claims that when firing the transitions and achieving at the reachable markings in N_3, from the first marker to the last marking, there is a shortest route. Figure 9 displays the key nodes generated in the execution process. The transition firing sequence in the last node is the min-cost one. Of course, during the execution of this algorithm, some temporary nodes are generated and then deleted since they are not on the shortest path. In Figure 9, in order to make the description concise and explicit, we ignore these non-critical nodes, but indeed they ever existed in the process.

According to Algorithm 6, the reachable marking of the last node is the final marking of the DDuM model, which means that the best alignment is discovered. The cost of this node is that of the optimal alignment, and the transition firing

Algorithm 6 Search for an optimal alignment between Petri nets and traces based on the DDuM model

Input: DDuM model $N_{\mathrm{D}} = (P_{\mathrm{D}}, T_{\mathrm{D}}; F_{\mathrm{D}}, \alpha_{\mathrm{D}}, m_{\mathrm{iD}}, m_{\mathrm{fD}})$;
Output: An optimal alignment γ of the DDuM model N_{D}.

1: $queue \leftarrow \emptyset$ // to initialize a queue to store the reachable markings and cost of the DDuM model
2: $node.c \leftarrow 0$; $node.m \leftarrow m_{\mathrm{iD}}$; $node.\gamma \leftarrow <>$ // *node* stands for the element in the queue, c is the current cost, m is the current reachable marking, γ is the current transition firing sequence
3: $queue \leftarrow queue \cup \{node\}$;
4: **while** $queue \neq \emptyset$ **do**
5: search for the node *node* with the minimal cost;
6: **if** $node.m \neq m_{\mathrm{fD}}$ **then**
7: **for** $\forall t_i \in T_{\mathrm{D}}$ **do**
8: **if** $(t_i) \subseteq node.m$ **then**
9: $newnode.m \leftarrow node.m \cup (t_i) - (t_i)$; // to generate a new node
10: **if** $(\pi_1(t_i) \neq >>$ AND $\pi_2(t_i) \neq >>)$ **OR** $(\alpha_{\mathrm{D}}(t_i) \neq \tau)$ **then**
11: $newnode.c \leftarrow node.c$; // to remain the cost when synchronous/invisible transition is fired
12: **else**
13: $newnode.c \leftarrow node.c + 1$ // to increase the cost by 1 when log/model transition is fired
14: **end if**
15: $newnode.\gamma \leftarrow node.\gamma \oplus < t_i >$;
16: **if** $\exists oldnode \in queue$ AND $oldnode.m = newnode.m$ **then**
17: **if** $oldnode.c \geq newnode.c$ **then**
18: $queue \leftarrow queue - \{oldnode\}$; // to remove the existing node from the queue
19: $queue \leftarrow queue \cup \{newnode\}$ // to add the new node to the queue
20: **end if**
21: **else**
22: $queue \leftarrow queue \cup \{newnode\}$; // to add the new node to the queue
23: **end if**
24: **end if**
25: **end for**
26: $queue \leftarrow queue - \{node\}$
27: **else**
28: map the transition sequence $newnode.\gamma$ to the move sequence γ;
29: **return** γ; // to return the optimal alignment
30: **end if**
31: **end while**

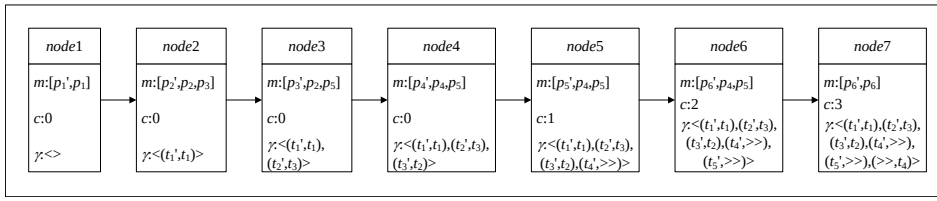


Figure 9. The process of searching for an effective shortest path

sequence of this node is the transition sequence that can be mapped to the optimal alignment. As shown in Figure 9, given $S_1, lc((a,t)), \sigma_1$ and N_1, the cost of the optimal alignment is 3, and the transition sequence corresponding to the optimal alignment is $\langle(t_1', t_1), (t_2', t_3), (t_3', t_2), (t_4' >>), (t_4', >>), (>>, t_4)\rangle$. By mapping the transitions of the first column of the firing sequence to the activities, we can get an optimal alignment $\gamma = \langle(a, t_1), (c, t_3), (b, t_2), (e, >>), (f, >>), (>>, t_4)\rangle$. Through alignment γ, the deviations $(e, >>), (f, >>)$ and $(>>, t_4)$ can be detected and located.

In addition, Algorithm 6 may enter an endless loop when there is a loop in which all the transitions are the invisible ones in the Petri net. In this case, the effective solution is to modify the standard likelihood cost function to assign a small value greater than 0 to the invisible moves as their cost. However, this cost should be far less than that of the log and model moves, such as 0.1, etc., which can avoid abandoning invisible transitions and choosing log transitions or model transitions during alignment. Thus, the new cost function will not cause a fatal impact on the search of the effective path.

4 EXPERIMENTAL EVALUATIONS

Given the process model, trace, milestone set and cost function, a DDuM model and an optimal alignment can be obtained. According to optimal alignments, all the deviations can be detected and located. Among many conformance checking methods, the MTCG method is the unique one that considers the impact of milestone activities on the alignment. However, this method needs to calculate the reachable marking graphs of Petri nets, which means that it cannot calculate the alignment of complex models and has high complexity. Hence, the MTCG method has some limitations for the alignment between traces and process models.

In addition, the discrepancy between the DDuM method and other traditional alignment methods is that it takes into account the influence of milestone activities on the alignment process and results. Specifically, when there are 0 milestone activities, that is, when no milestone activities are specified, the DDuM method does not need to perform the synchronous composition of the transitions labeled with the milestone activities. It is only necessary to compare the activities in the log and the

Petri net, choose the transitions labeled with the same activities and then perform the product of these transitions. In the circumstance, the execution process of the DDuM method is the same as that of A* alignment algorithm. Hence, A* alignment algorithm is a special case of the DDuM method.

Research is done on a clinical process of diabetic foot ulcer during the COVID-19 epidemic and a set of logs. Our experiment using the Java language was accomplished on a computer with an Intel® Core™ i7-8750H processor, a 64-bit Windows 10 system, 16.0 GB RAM and JDK 1.8. According to the standard clinical process of diabetic foot ulcer during the COVID-19 epidemic, a process model is manually created using Petri nets. In the model, choice, sequential, and parallel structures are all included. And there are 49 transitions, 52 places, and 116 arcs in the Petri net. Each transition is associated with an activity within the medical process.

In the real-life executions of the clinical process, some activities may overlap or some may be recorded twice. This leads to the event logs deviating from the given model Nd. According to the Petri net, the entire fit traces with various numbers of activities are the entire fit traces with various activity counts are created, and there are 36 336 different fit traces. There are 48, 576, 4 608, 1 152, 11 520 and 18 432 fit traces with the lengths of 9, 25, 29, 30, 34 and 38, respectively.

The incomplete fit traces are formed by introducing noise. Noise is introduced by randomly deleting or adding activities in the trace. A formula $noise = d/|\sigma_{\text{fit}}|$ can be used to calculate the noise ratio, where there are d deviations which are the inserted or deleted activities, and $|\sigma_{\text{fit}}|$ is the original fit trace's length.

The results of this experiment are introduced in two aspects as follows:

1. For the traces with different lengths, the mean number of queued states and computation time for deviation detection are compared between different milestone-based approaches. There are two types of event logs in use. One category consists of correctly fitting traces with varying lengths, while the other comprises incorrectly fitting traces with varying lengths. The unfit traces' noise ratios are randomly between 5 % and 30 %.

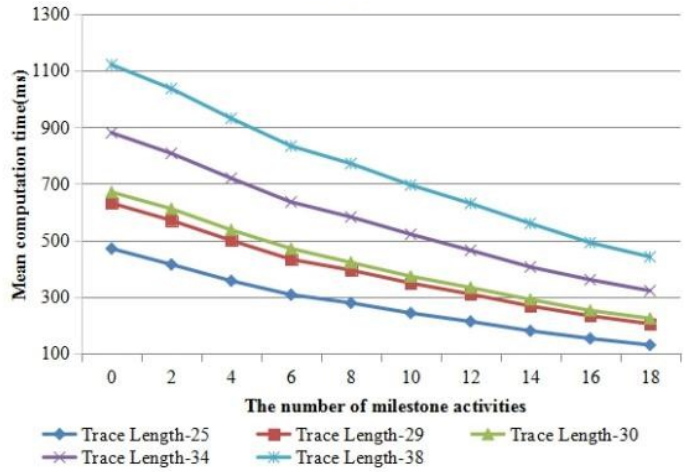
2. Regarding traces with varying noise ratios, mean computation time, and number of queued states for deviation detection are compared among milestone-based approaches. There are two types of event logs in use. One includes the traces with lengths from 25 to 29 and the other includes the traces with lengths from 34 to 38. The noise ratio of the traces in each log is 0 %, 5 %, 10 %, 15 %, 20 %, 25 % or 30 %, respectively.

The trace length refers to the number of activities in the fit trace denoted by $|\sigma_{\text{fit}}|$, but not the actual length of the trace.

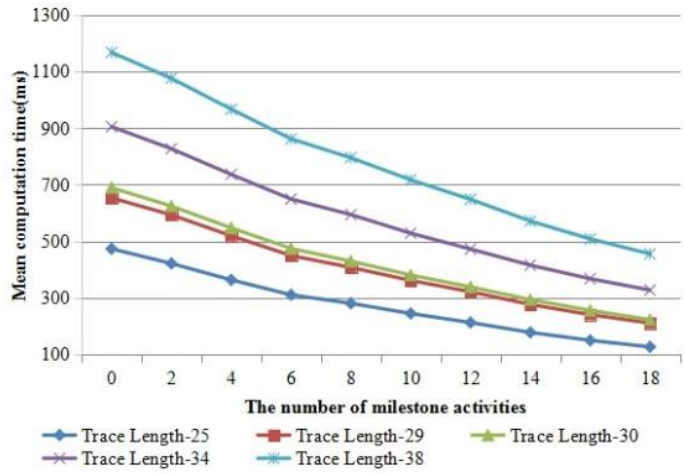
There are 24 event logs in this experiment, and each log includes 100 traces. All the data are the average results that are recorded by the same experiments are repeated for 10 times.

4.1 Trace Length

In this example, 10 event logs are used, each of which contains 100 traces. The average length of traces in the 10 logs are 25, 29, 30, 34 and 38, respectively. When there are 0, 2, 4, 6, 8, 10, 12, 14, 16 and 18 milestone activities, the comparisons of queued states and mean computation time for detecting the deviations are shown in Figures 10 and 11, respectively.

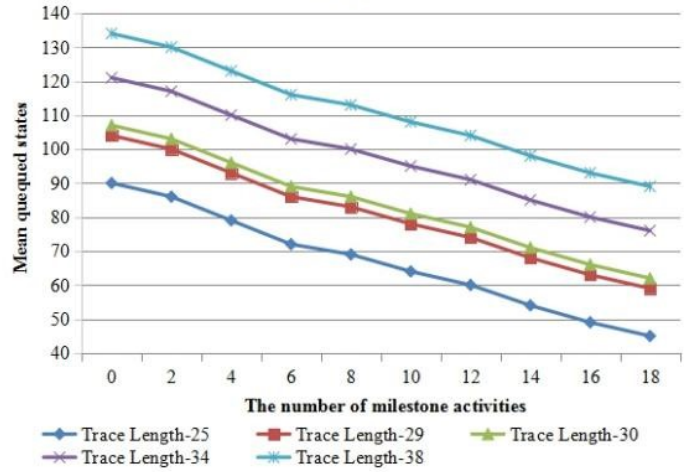


a) Fit trace

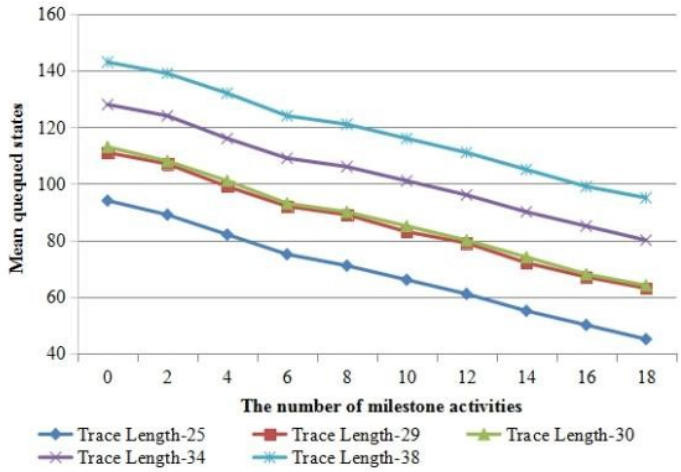


b) Unfit trace

Figure 10. Comparison of mean computation time for DDuM with different milestones among various trace lengths



a) Fit trace



b) Unfit trace

Figure 11. Comparison of mean queued states for DDuM with different milestones among various trace lengths

As shown in Figure 10, when milestone activities remain constant, the longer the trace length is, the longer the mean computation time is. Meanwhile, when the average lengths of traces are equal, as the specified milestone activities increases, the mean computation time for deviation detection gradually decreases and its change becomes smaller and smaller.

Through the analysis of the data in Figure 10, the DDuM models are more and more complex as milestone activities or the trace length increase. The DDuM

models are the search space for deviation detection, and the comprehensive process is shown in Algorithm 6. With the decrease of the search space, the efficiency of calculating the optimal alignment will be improved. Usually, the DDuM models effect the execution time of Algorithm 6. The more complicated the models are, the longer the mean computation time for deviation detection is.

When comparing the information presented in Figure 10 a) and Figure 10 b), the change of the fit trace set is similar to that of the unfit set. However, the data of the fit trace is slightly less than that of the unfit set, which deviates from the comparison results of the DDuM model. Generally, the deviations for the unfit trace set is larger than that for the fit trace set, so the computation time of the fit trace set is shorter than that of the unfit trace set.

As shown in Figure 11, mean queued states for the deviation detection are recorded and compared when the trace length and milestone activities vary. The change of mean queued states conform to the mean computation time, as illustrated in Figure 10. The reason that causes to the comparison result of queued states is also similar to that of mean computation time.

In conclusion, when the deviations between the same trace and Petri net are detected based on the given cost function, the more the specified milestone activities are, the smaller the search spaces generated by the DDuM method are, the less complex the procedures for the deviation detection is. When there are 0 specified milestone activities, the DDuM method degenerates to A* alignment method.

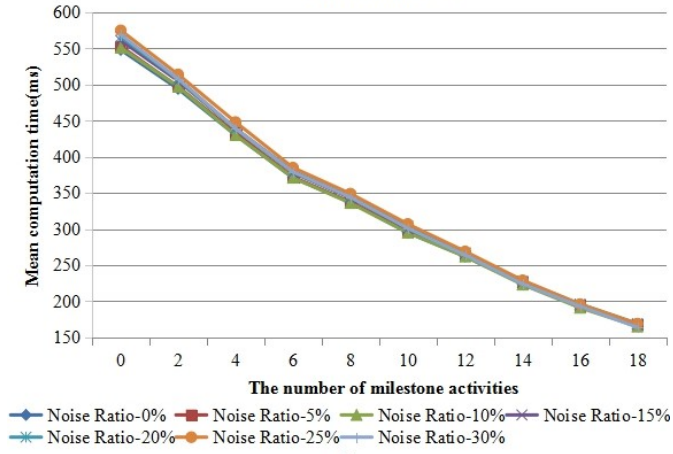
4.2 Noise Level

In this subsection, the given event logs are divided into two different sets according to the trace lengths: Trace Length 25–29, in which the trace length is between 25 and 29, and Trace Length 34–38, in which the trace length is from 34 to 38. All the comparisons with different milestones between various noise ratios are shown in Figures 12 and 13.

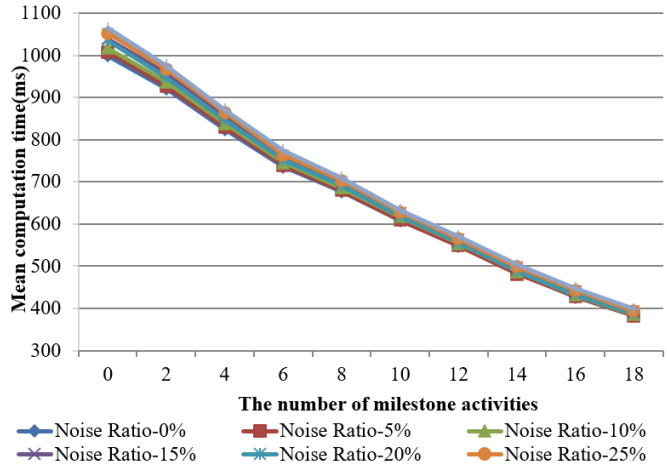
Referring to Figure 12, when the average trace lengths are identical, the mean computation time just varies with the milestones. The comparison results are similar to those in Figure 11. Referring to Figure 13, when the average trace lengths are identical, the mean queued states obviously vary with the milestones, and are slightly influenced by the noise ratios. The higher the noise ratios are, the higher the queued states are. But the decrease rates become slower along with the growth of the queued states. The comparison results are similar to those in Figure 11.

After comparing and analyzing all the experimental data, we can draw two conclusions:

1. The influence of the trace length on the complexity of the DDuM method is greater than that of the noise ratio.



a) Trace length 25–29

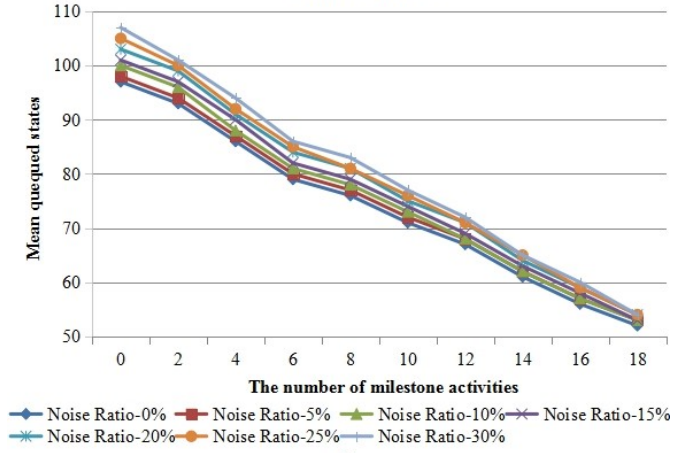


b) Trace length 34–38

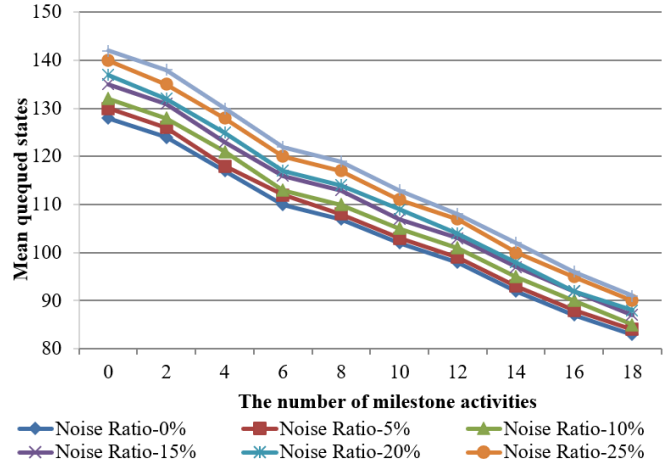
Figure 12. Comparison of mean computation time for DDuM with different milestones among various noise ratios

2. The complexity of the DDuM method greatly decreases as the milestones rises. When milestones is 0, the DDuM method is the same as the A* alignment method.

Hence, the DDuM method outperforms A* alignment method.



a) Trace length 25–29



b) Trace length 34–38

Figure 13. Comparison of mean queued states for DDuM with different milestones among various noise ratios

5 CONCLUSIONS

The introduction of milestone activities in business processes brings new hope for improving the efficiency of alignment. The milestone activities can not only effectively remove the noise in the event log, but also further reasonably restrict the firing of transitions in the process model. This paper proposes an alignment method of business process with milestone activities, which is called DDuM. Through binding the milestone activities in the case and Petri net to synchronously occur, the effectiveness of calculating the optimal alignment is greatly improved.

In the future, this approach will be further analyzed and applied to expand its application field. The work to be carried out mainly includes:

1. The properties of the DDuM model need to be further studied, such as free deadlock, boundedness, and liveness;
2. The method can be further optimized, since the process model can be preprocessed and the branches without milestone activities can be pruned;
3. A new algorithm can be proposed based on the DDuM model to calculate all the optimal alignments between the case and Petri net.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant No. 72101137 and Grant No. 61973180, in part by the Natural Science Foundation of Shandong Province under Grant No. ZR2019MF033, Grant No. ZR2020MF033 and Grant No. ZR2021MF117, in part by the Education Ministry Humanities and Social Science Research Youth Fund Project of China under Grant No. 20YJCZH159 and Grant No. 21YJCZH150, in part by the Shandong Key R & D Program (Soft Science) Project under Grant No. 2022RKY02009, and in part by the "Taishan Scholar" Construction Program of Shandong Province, in part by the Shandong Digital Economy Research Base Project of the Research Center of Shandong Province on "Xi Jinping Thought on Socialism with Chinese Characteristics for a New Era" and Shandong University of Science and Technology under Grant No. SDSZJD202314.

REFERENCES

[1] Issahaku, F. Y.—Lu, K.—Fang, X.—Danwana, S. B.—Bandago, H. M.: An Overview of Semantic-Based Process Mining Techniques: Trends and Future Directions. Knowledge and Information Systems, Vol. 66, 2024, No. 10, pp. 5783–5827, doi: 10.1007/s10115-024-02147-x.

[2] van Zelst, S. J.—Mannhardt, F.—De Leoni, M.—Koschmider, A.: Event Abstraction in Process Mining: Literature Review and Taxonomy. Granular Computing, Vol. 6, 2021, No. 3, pp. 719–736, doi: 10.1007/s41066-020-00226-2.

[3] Akhramovich, K.—Serral, E.—Cetina, C.: A Systematic Literature Review on the Application of Process Mining to Industry 4.0. Knowledge and Information Systems, Vol. 66, 2024, No. 5, pp. 2699–2746, doi: 10.1007/s10115-023-02042-x.

[4] Martin, N.—Fischer, D. A.—Kerpedzhiev, G. D.—Goel, K.—Leemans, S. J. J.—Röglinger, M.—van der Aalst, W. M. P.—Dumas, M.—La Rosa, M.—Wynn, M. T.: Opportunities and Challenges for Process Mining in Organizations: Results of a Delphi Study. Business & Information Systems Engineering, Vol. 63, 2021, No. 5, pp. 511–527, doi: 10.1007/s12599-021-00720-0.

[5] Suriadi, S.—Andrews, R.—ter Hofstede, A. H. M.—Wynn, M. T.: Event Log Imperfection Patterns for Process Mining: Towards a Systematic Approach to Cleaning Event Logs. Information Systems, Vol. 64, 2017, pp. 132–150, doi: 10.1016/j.is.2016.07.011.

[6] Ghasemi, M.—Amyot, D.: From Event Logs to Goals: A Systematic Literature Review of Goal-Oriented Process Mining. Requirements Engineering, Vol. 25, 2020, No. 1, pp. 67–93, doi: 10.1007/s00766-018-00308-3.

[7] Jagadeesh Chandra Bose, R. P.—van der Aalst, W. M. P.: Process Diagnostics Using Trace Alignment: Opportunities, Issues, and Challenges. Information Systems, Vol. 37, 2012, No. 2, pp. 117–141, doi: 10.1016/j.is.2011.08.003.

[8] Shen, X.—Liu, C.—Li, H.—Zheng, K.—Cheng, L.—Zeng, Q.: Distributed Conformance Checking Method Based on Process Model Decomposition. Computer Integrated Manufacturing Systems, Vol. 30, 2024, No. 8, pp. 2884–2896 (in Chinese).

[9] Sun, H.—Du, Y.—Qi, L.—He, Z.: A Method for Mining Process Models with Indirect Dependencies via Petri Nets. IEEE Access, Vol. 7, 2019, pp. 81211–81226, doi: 10.1109/ACCESS.2019.2923624.

[10] Li, J.—Yang, R.—Ding, Z.—Pan, M.: A Method for Learning a Petri Net Model Based on Region Theory. Computing and Informatics, Vol. 39, 2020, No. 1–2, pp. 174–192, doi: 10.31577/cai_2020_1-2_174.

[11] Liu, C.—Duan, H.—Zeng, Q.—Zhou, M.—Lu, F.—Cheng, J.: Towards Comprehensive Support for Privacy Preservation Cross-Organization Business Process Mining. IEEE Transactions on Services Computing, Vol. 12, 2019, No. 4, pp. 639–653, doi: 10.1109/TSC.2016.2617331.

[12] Li, M.—Ding, Z.: A Preface to the Special Issue on Emerging and Intelligent Information Services. Computing and Informatics, Vol. 39, 2020, No. 1–2, pp. 1–4, doi: 10.31577/cai_2020_1-2_1.

[13] Mozafari Mehr, A. S.: Multi-Perspective Conformance Checking: Identifying and Understanding Patterns of Anomalous Behavior. Ph.D. Thesis. Eindhoven University of Technology, 2024.

[14] De Leoni, M.—Maggi, F. M.—van der Aalst, W. M. P.: Aligning Event Logs and Declarative Process Models for Conformance Checking. In: Barros, A., Gal, A., Kindler, E. (Eds.): Business Process Management (BPM 2012). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 7481, 2012, pp. 82–97, doi: 10.1007/978-3-642-32885-5_6.

[15] Adriansyah, A.—Van Dongen, B. F.—van der Aalst, W. M. P.: Towards Robust Conformance Checking. In: Muehlen, M., Su, J. (Eds.): Business Process

Management Workshops (BPM 2010). Springer, Berlin, Heidelberg, Lecture Notes in Business Information Processing, Vol. 66, 2011, pp. 122–133, doi: 10.1007/978-3-642-20511-8_11.

[16] Bloemen, V.—van Zelst, S.—van der Aalst, W.—van Dongen, B.—van de Pol, J.: Aligning Observed and Modelled Behaviour by Maximizing Synchronous Moves and Using Milestones. Information Systems, Vol. 103, 2022, Art. No. 101456, doi: 10.1016/j.is.2019.101456.

[17] Pauwels, S.—Calders, T.: An Anomaly Detection Technique for Business Processes Based on Extended Dynamic Bayesian Networks. Proceedings of the 34th ACM SIGAPP Symposium on Applied Computing (SAC '19), 2019, pp. 494–501, doi: 10.1145/3297280.3297326.

[18] Vertuam Neto, R.—Tavares, G.—Ceravolo, P.—Barbon, S.: On the Use of Online Clustering for Anomaly Detection in Trace Streams. Proceedings of the XVII Brazilian Symposium on Information Systems (SBSI '21), ACM, 2021, doi: 10.1145/3466933.3466979.

[19] Nguyen, H.—Dumas, M.—La Rosa, M.—Maggi, F. M.—Suriadi, S.: Business Process Deviance Mining: Review and Evaluation. 2016, doi: 10.48550/arXiv.1608.08252 (arXiv Preprint arXiv:1608.08252).

[20] Vijayakamal, M.—Vasumathi, D.: A Novel Approach to Detect Anomalies in Business Process Event Logs Using Deep Learning Algorithm. In: Reddy, V. S., Prasad, V. K., Wang, J., Reddy, K. T. V. (Eds.): Soft Computing and Signal Processing (ICSCSP 2020). Springer, Singapore, Advances in Intelligent Systems and Computing, Vol. 1340, 2022, pp. 363–374, doi: 10.1007/978-981-16-1249-7_34.

[21] Lahann, J.—Pfeiffer, P.—Fettke, P.: LSTM-Based Anomaly Detection of Process Instances: Benchmark and Tweaks. In: Montali, M., Senderovich, A., Weidlich, M. (Eds.): Process Mining Workshops (ICPM 2022). Springer, Cham, Lecture Notes in Business Information Processing, Vol. 468, 2023, pp. 229–241, doi: 10.1007/978-3-031-27815-0_17.

[22] Elaziz, E. A.—Fathalla, R.—Shaheen, M.: Deep Reinforcement Learning for Data-Efficient Weakly Supervised Business Process Anomaly Detection. Journal of Big Data, Vol. 10, 2023, No. 1, Art. No. 33, doi: 10.1186/s40537-023-00708-5.

[23] Tian, Y.—Yang, L.—Han, D.—Du, Y.: Conformance Checking Method of Business Processes Based on Improved BERT and Lightweight CNN. Computer Engineering, Vol. 51, 2025, No. 7, pp. 199–209, doi: 10.19678/j.issn.1000-3428.0069357 (in Chinese).

[24] Cook, J. E.—Wolf, A. L.: Software Process Validation: Quantitatively Measuring the Correspondence of a Process to a Model. ACM Transactions on Software Engineering and Methodology (TOSEM), Vol. 8, 1999, No. 2, pp. 147–176, doi: 10.1145/304399.304401.

[25] Lu, X.—Fahland, D.—van der Aalst, W. M. P.: Conformance Checking Based on Partially Ordered Event Data. In: Fournier, F., Mendling, J. (Eds.): Business Process Management Workshops (BPM 2014). Springer, Cham, Lecture Notes in Business Information Processing, Vol. 202, 2015, pp. 75–88, doi: 10.1007/978-3-319-15895-2_7.

[26] Feng, X.—Han, D.—Tian, Y.: Analysis and Application of Min-Cost Transition Systems to Business Process Management. Computing and Informatics, Vol. 39, 2020, No. 1–2, pp. 213–245, doi: 10.31577/cai_2020_1-2_213.

[27] Tian, Y.—Pang, X.—Yang, R.—Han, D.—Wang, L.—Du, Y.: Method for Predicting the Absolute Remaining Time of Business Processes Based on Prefix Trace Representation Learning and Attention Mechanism. Computer Integrated Manufacturing Systems, Vol. 31, 2025, No. 5, pp. 1762–1778, doi: 10.13196/j.cims.2024.BPM12 (in Chinese).

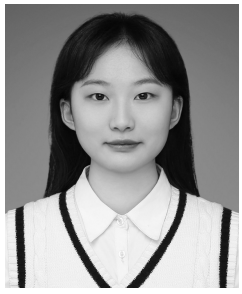
[28] Wu, W.—Xing, Z.—Yue, H.—Su, H.—Pang, S.: Petri-net-based Deadlock Detection and Recovery for Control of Interacting Equipment in Automated Container Terminals. IET Intelligent Transport Systems, Vol. 16, 2022, No. 6, pp. 739–753, doi: 10.1049/itr2.12168.

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

[30] Tian, Y.—Li, X.—Wu, Y. H.—Han, D.—Du, Y.—Wang, L.: Batch Repair of Event Logs Based on Constrained Trace Clustering. Computer Integrated Manufacturing Systems, Vol. 30, 2024, No. 8, pp. 2797–2808, doi: 10.13196/j.cims.2023.BPM11 (in Chinese).

Yinhua Tian received her B.Sc. degree in computer science and technology, her M.Sc. degree and her Ph.D. degree in computer software and theory from the Shandong University of Science and Technology, Qingdao, China, in 2004, 2007 and 2018, respectively. She is currently Associate Professer of computer science and technology with the Shandong University of Science and Technology. She has authored over 20 technical papers in journals and conference proceedings. Her current research interests include Petri nets, process mining, and optimization algorithms.

Ruizhe Zhang received his B.Sc. degree from the Shandong University of Science and Technology, Jinan, China, in 2023. He is currently pursuing his M.Sc. degree with the College of Intelligence Equipment, Shandong University of Science and Technology, Tai'an, China. His main research interests include process mining and predictive process monitoring.



Wen Liu received her B.Sc. degree from the Shandong University of Science and Technology, Tai'an, China, in 2024. She is currently pursuing her M.Sc. degree with the College of Intelligence Equipment, Shandong University of Science and Technology, Tai'an, China. Her main research interests include process mining and predictive process monitoring.

Yuyue Du received his B.Sc. degree from the Shandong University, Jinan, China, in 1982, his M.Sc. degree from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 1991, and his Ph.D. degree in computer application from the Tongji University, Shanghai, China, in 2003. He is currently Professor at the College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao, China. He has taken in over 10 projects supported by the National Nature Science Foundation, the National Key Basic Research Developing Program, and other important and key projects at provincial levels. He has published over 240 papers in domestic and international academic publications, and they are embodied over 80 times by SCI and EI and cited over 930 times by others. His research interests are in formal engineering, Petri nets, real-time systems, Web services, and workflows. He is a member of the Professional Committee of Petri Nets of the China Computer Federation.