Analysis of Business Processes with Automatic Detection of KPI Thresholds and Process Discovery Based on Trace Variants

Taro Takei[1], Hiroki Horita[2*]

[1]Ibaraki University, Hitachi, Ibaraki 316–8511, Japan

[2]Ibaraki University, Hitachi, Ibaraki 316–8511, Japan, hiroki.horita.is@vc.ibaraki.ac.jp

Received: July 11, 2023; Accepted: August 02, 2023; Published: September 05, 2023

Abstract

One effective way to analyze an event log of a complex business process is to filter the event log by a KPI threshold and then use an analysis algorithm. Event logs filtered by KPI thresholds are simpler and easier to understand than the original business process. KPI thresholds have conventionally had to be set through a trial-and-error process, but existing research has proposed an automatic KPI threshold detection method that reduces the time and effort required to search for a threshold value. In the existing method, an event log is first divided by an arbitrary number k, and a business process model is generated for the divided event log using a process discovery algorithm. By repeatedly aggregating similar business process models, it is possible to analyze how the business process models differ according to KPI values. However, the existing method requires trial-and-error to determine the value of k because the detection threshold and business process model vary depending on the value of k, which is time-consuming. Therefore, this paper proposes a method to automatically detect KPI thresholds by dividing event logs based on trace variants. Experimental results show that the business process models detected by the proposed method are simpler and the threshold detection time is significantly reduced.

Keywords: data mining; process mining; process discovery; graph edit distance.

1 Introduction

Process mining is a technology for visualizing and analyzing processes by extracting valuable information from event logs of business processes [1]. Data-driven business analysis has flourished in recent years [2]. Many machine learning methods are inadequate to identify and explain various problems in business processes. Process mining for discovering, monitoring, and improving business processes can be particularly useful for understanding business processes. However, as the size of organizations increases, business processes become more complex, and the business processes discovered by process mining techniques become so-called spaghetti processes that are difficult to analyze [1].

As a method for analyzing complex business processes, event logs can be filtered by focusing on Key Performance Indicators (KPIs) to extract only specific processes. There is a lot of research on analyzing business processes using KPIs [3-12], and by setting a KPI threshold, it is possible to extract and analyze only those processes that are of interest to the user. For example, if only event logs are extracted during busy periods, it is possible to analyze how they differ from normal processes. The problem here is that when filtering event logs focusing on a specific KPI, it is necessary to go through a trial-and-error process to determine the KPI threshold value. For example, if we try to extract two

business process models by splitting the event logs into those with execution times of less than 5 minutes and those with execution times of more than 5 minutes, there may be no difference between the two. In such cases, it is necessary to perform a trial-and-error process such as setting the threshold value in one-minute increments and extracting the business process until a difference is found in the extracted business process model. Since such trial-and-error work is an obstacle to analysis, Abe et al. proposed an approach to automatically detect KPI thresholds from event logs [13]. This approach reduces the effort required for users to set KPI thresholds.

Abe et al.'s method sorts of process instances based on KPI values and divides them into k groups. Next, a process model is generated for each group of k equally divided process instances, and the graph edit distance between each group is calculated. Finally, the process models with the smallest graph edit distance, i.e., the most similar process models, are aggregated into one, and this process is repeated until all edit distances are constant. The groups of similar process models and their respective KPI thresholds are then calculated. However, in the existing method by Abe et al., the threshold detected and the process model change depending on the initial number of log divisions k, so it is necessary to change the value of k by the user's trial. The user needs to set the value of k by trial-and-error. If the value of k is increased too much, the computation time for similarity increases explosively. If k is too small, it will cause a mixing of dissimilar process instances, resulting in the detection of inappropriate threshold values.

In this paper, we propose a method for detecting KPI thresholds by dividing event logs based on trace variants. By splitting the event log based on trace variants, we can avoid trial-and-error in determining k and avoid mixing dissimilar business processes. In addition, grouping by trace variants simplifies the business process of each group, reduces the computation time for calculating similarity based on graph edit distance, and reduces the threshold detection time. Simplified business processes are easier for users to understand, and the reduced detection time of KPI thresholds enables more efficient analysis of the business process.

To evaluate the effectiveness of the proposed method, we conducted experiments to compare the existing method with the proposed method using synthetic event logs and real-life event logs. In the experiment using synthetic event logs, first, synthetic logs with a small number of variants and a large average trace size are prepared. Next, we measured the threshold calculation time for each value of the number of divisions k using the existing method and compared the results with the two synthetic logs. Similarly, we perform threshold detection using the proposed method on the two synthetic logs and measure the computation time. In an experiment using real-life event logs, we compared the KPI thresholds, process models, and threshold calculation times of the existing and proposed methods using a dataset of a travel expense claim process that was executed at Eindhoven University of Technology (TU/e).

The remainder of this paper is organized as follows. Section 2 describes the knowledge and techniques related to this study. Section 3 then introduces the related works. Section 4 describes our proposed method for automatic KPI threshold detection and process discovery based on trace variants. In section 5, we apply the existing and proposed methods to synthetic and real-life event logs and compare the results. Finally, section 6 summarizes this paper and discusses future work.

2 Background knowledge

This section describes the knowledge associated with the proposed method.

2.1 Process Mining

Process mining is a technique for extracting and utilizing useful information from event logs output by information systems. There are three main types of process mining technologies, which are classified as process discovery, conformance checking, and process enhancement [1].

2.1.1 Process Discovery

Process Discovery is a technology for automatic construction of business process models from an event log. Since process discovery is the first step in generating a business process model from an event log, it can be regarded as the core technology of process mining. The inductive mining [14] used in this paper generates a process tree equivalent to a Petri net from an event log. Process trees can be converted to Petri nets, BPMN models, etc. Furthermore, process trees generated by inductive mining have high fitness and guaranteed soundness since all behaviors recorded in an event log can always be replayed [1]. A process model with high soundness guarantees that a place does not have multiple tokens (safety), that the process terminates properly (normal termination), that the process can always be terminated (optional termination), and that there are no dead transitions.

2.1.2 Event Log

Event log is structured data in which multiple cases (traces) are characterized by various attributes such as activity, timestamp, resource, and cost. The meaning of each attribute is as follows: activity is the name of the event in the process, timestamp is the start and end time of the activity,

resource is the source of the activity, and cost is the cost associated with the activity, and each is associated with each case as a number or a string.

An example of a typical event log is shown in Table 1. Main event log file formats used in process mining are CSV (Comma Separated Values) and XES (eXtensible Event Stream) [15]. All these formats are supported by a process mining software called ProM .

Table 1. Example of event logs

Case ID	Activity	timestamp	Cost
0	Activity A	1970-01-01 09:00:00+09:00	100
0	Activity AG	1970-01-01 09:26:22+09:00	200
0	Activity AI	1970-01-01 09:42:46+09:00	300
0	Activity AJ	1970-01-01 10:07:41+09:00	100
0	Activity AI	1970-01-01 10:31:56+09:00	300
0	Activity AH	1970-01-01 11:00:53+09:00	0
0	Activity B	1970-01-01 11:29:08+09:00	050
1	Activity A	1970-01-01 09:00:00+09:00	100
1	Activity C	1970-01-01 09:21:10+09:00	50
1	Activity Q	1970-01-01 09:40:12+09:00	200

2.2 Graph Edit Distance

Graph edit distance is the minimum cost of editing (transforming) one graph into the other [16]. There are three basic types of operations in graph transformation.

- Replace node: Operation to replace a node in one graph with a node in the other graph
- Insert/Remove Node: Inserts a node into or removes a node from a graph
- Insert/Remove Edge: Inserts an edge into or removes an edge from a graph

Each transformation operation has a cost given by a cost function. When transforming one graph into the other, the graph edit distance algorithm searches for all possible combinations of transformation operations and returns the set that minimizes the total cost of the transformation operations [16]. Thus, the smaller the total cost of the transformation operations, the fewer operations are required to transform the graphs, i.e., the higher the similarity between the graphs. The details of the graph edit distance used in this paper are explained in section 4.1.4.

3 Related Works

This section presents related works to this paper. Most like our paper is the paper by Abe et al. [13]. Abe et al. [13] proposed a method to automatically calculate the threshold based on the graph edit distance between process models to reduce the time and effort of trial and error for KPI thresholds.

Graph edit distance is the minimum cost of transforming one graph into the other, as explained in Section 2.2, and can be used to quantify the structural similarity of the graphs. The general procedure of the KPI threshold detection method based on graph edit distance is as follows.

1. Write an external definition of the KPI formula and calculate the KPI value for each process instance (trace).
2. Sort traces based on KPI values and divide traces into k segments with an arbitrary number of k.
3. Generate a process model from the divided traces.
4. Calculate the graph edit distance between the generated process models, and extract KPI thresholds with large changes between processes by aggregating models with close distances.

Abe et al. used event logs generated from web application servers to detect KPI thresholds. In the event logs of the web application server, KPI thresholds were detected using the number of accesses to the help page and the time from process start to process end as KPIs. As a result, in the experiment focusing on the number of times the help page was accessed, the process models for the 6 groups were aggregated into 3 groups, and the similarity of the processes in which users accessed the help page more than 3 times was observed. the processes in which users accessed the help page more than 3 times were similar. In the experiment focusing on duration, the 10-group process model was aggregated to 6 groups, and the processes in the first and second groups were like the task. The processes in the first and second groups showed fewer transitions between tasks. These results show that the automatic detection method of KPI thresholds can be applied to real-world event logs, and that it is possible to reduce the time required to search for thresholds that can distinguish between processes. However, Abe et al. identified the following issues [13].

- It is necessary to consider how to determine and provide the appropriate number of k partitions.
- Need to consider how to eliminate redundant data to extract more valuable information from the log data.

- Need to consider how to automatically detect log semantics and KPI lists.
- It is also necessary to consider the analysis method when multiple types of KPIs are considered.

This study focuses on how to provide an arbitrary number of divisions k and proposes a method to automatically detect KPI thresholds by dividing logs based on trace variants.

4 Proposed Method

This section describes the proposed method, which is an improvement of the existing KPI threshold auto-detection method [13] to a trace variant-based method. Figure 1 shows an overview of the proposed method.

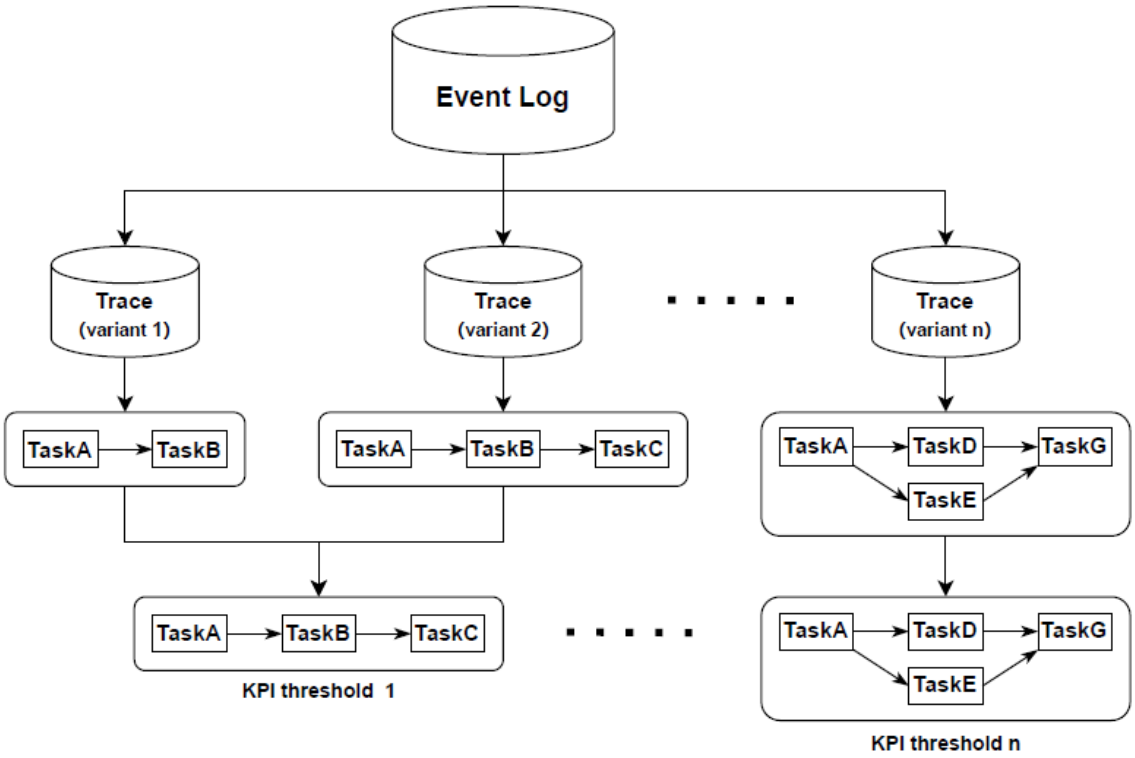


Figure 1. Schematic diagram of the proposed method

4.1 Event Log partitioning and Process Discovery Based on Trace Variants

In this section, we describe the procedure for event log partitioning and process discovery based on trace variants. The algorithm 1 is shown below. The event logs used as input for the proposed method must contain at least three attributes: case id, activity, and timestamp. The proposed method represents the process model as a Petri net.

Algorithm 1 Log partitioning and process discovery based on trace variants

INPUT: $p_0, ..., p_{n-1} \in L$: The event log L is composed of n process instances.
OUTPUT: Group of Petri nets $N_0, ..., N_{k-1}$and KPI thresholds

for $i \leftarrow 0, n-1$ **do**
  Calculate KPI value v_i for process instance p_i using predefined formulas
end for

Sort L based on KPI value v_i, variant, and size of the trace
Split $p_0, ..., p_{n-1} \in L$ into k groups by variant unit

for $j \leftarrow 0, k-1$ **do**
  Generate Petri net N_j from trace p_j.
end for
Save KPI thresholds for Petri Net $N_0, ..., N_{k-1}$

4.1.1 Calculation of KPI values for each trace

First, KPI values are calculated for each trace based on the values recorded for each event in the event log. A trace is a series of events arranged by time stamp and represented as an event sequence. A KPI value is a numeric or string representation of a defined KPI that is calculated and associated with each trace. The default KPI values that can be calculated in this method are 'Duration', which is the execution time of the trace. Duration' can be calculated from the difference between the time of the last activity executed and the time of the first execution in the trace. When calculating KPI values other than 'Duration', it is necessary to define new calculation formulas. For example, the cost required to execute each event could be considered as a KPI.

4.1.2 Sorting and partitioning of event log

Next, the event log is sorted based on the KPI value of the trace, the variant of the trace, and the size of the trace. The trace variant refers to the type of traces present in the input event log, and the trace size refers to the number of events in the trace. After sorting, traces of the same variant are partitioned into a group. Figure 2 shows an example of a sorted and partitioned event log. By sorting and partitioning each trace variant as shown in the figure 2, the similarity between variants can be calculated without mixing different variants, unlike the method of abe et al. [13].

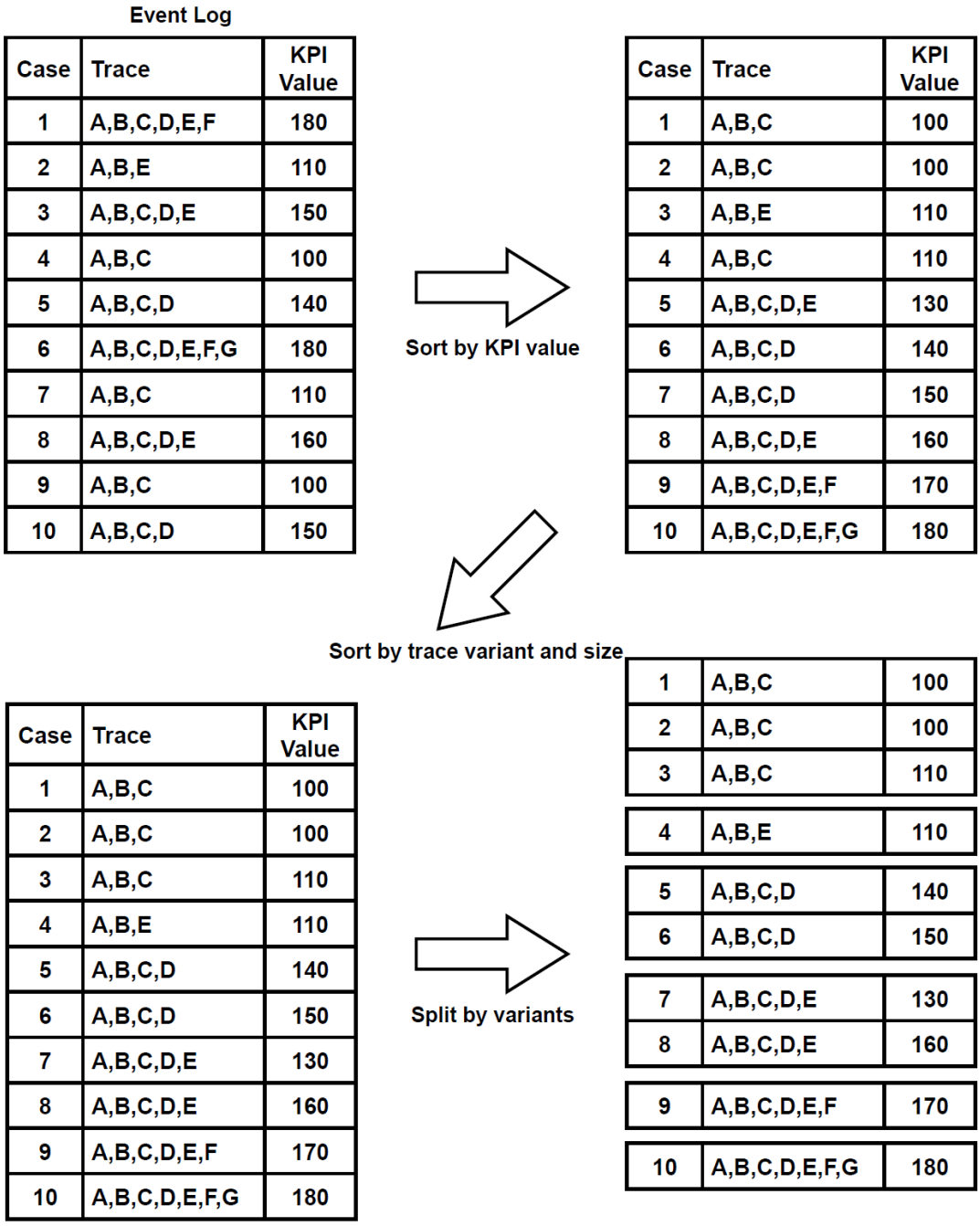


Figure 2. Example of event log partitioning

4.1.3 Process Discovery

For each group of partitioned traces, a Petri net is automatically generated using a process discovery algorithm. This method uses inductive mining [14] for process discovery. The explanation of inductive mining is given in section 2.1.1 and is therefore omitted here. After Petri nets are generated, the KPI threshold is stored for each group of Petri nets.

4.1.4 Automatic detection of KPI thresholds based on graph edit distance between process models

Finally, the similarity based on the graph edit distance is calculated for all combinations of the generated Petri nets, and the pair of Petri nets that are most similar are aggregated. In this study, the algorithm for graph aggregation based on the graph edit distance proposed in the existing method [13] was partially modified to be able to aggregate Petri nets. The modified algorithm based on the existing method, Algorithm 2, is shown below.

Algorithm 2 Aggregating Petri nets using similarity based on graph edit distance

Require: Petri Net group $N_0, ..., N_{k-1}$
Ensure: Aggregated Petri Net groups $N'_0, ..., N'_{k-1}$ and their KPI value domain conditions
1: Calculate similarity based on graph edit distance $d_{00}, d_{01}, ..., d_{ab}, ...d_{(k-1)(k-1)}$ for all Petri net combinations and input into table
2: **while** until all similarities fall below a certain level. **do**
3: Aggregate pairs of Petri nets with the greatest similarity N_a, N_b into a single Petri net $\{N_a, N_b\}$
4: Delete entries N_a and N_b from the similarity table
5: Generate new Petri nets $\{N_a, N_b\}$
6: Recalculate Petri net similarity $d_{00}, d_{01}, ..., d_{ab}, ...d_{(k-1)(k-1)}$ and update table
7: **end while**

In this study, we calculate the similarity between Petri nets using "Calculate Graph Edit Distance Similarity", which is implemented as a plug-in in an open-source software for process mining called ProM. The plugin is implemented based on the work of Remco Dijkman et al. [16] on evaluating graph matching algorithms and can compute the similarity between two Petri nets. The following is the similarity based on graph edit distance.

$G_1 = (N_1, E_1, \lambda_1)$and $G_2 = (N_2, E_2, \lambda_2)$ are graphs, respectively. M:N_1↛N_2be a partial injective mapping that maps nodes in G_1 to nodes in G_2, and let subn, skipn, and skip be the sets of substituted nodes, inserted or deleted nodes, and inserted or deleted edges. Furthermore, let 0≤wsubn≤1,0≤wskipn≤1,and 0≤wskipe≤1 be the weights assigned to node substitution, node insertion or deletion, and edge insertion or deletion, respectively. The percentage of inserted or deleted nodes fskipn, the percentage of inserted or deleted edges fskipe, and the average distance fsubsn of substituted nodes are defined as follows [16].

$$skipn = \frac{|skipn|}{|N_1|+|N_2|} \quad fskipe = \frac{|skipe|}{|E_1|+|E_2|} \quad fsubn = \frac{2.0 \cdot \sum_{(n,m)\in M} 1.0 - Sim(n,m)}{|subn|} \tag{1}$$

The graph edit similarity induced by the mapping M is as follows.

$$1.0 - \frac{wskipn \cdot fskipn + wskipe \cdot fskipe + wsubn \cdot fsubn}{wskipn + wskipe + wsubn} \tag{2}$$

The similarity between Petri nets is calculated using the formula 2, and the Petri nets that are most similar are aggregated. The Petri net aggregation process combines two Petri nets into one by merging the traces that generated the Petri nets into one and then performing process discovery on the merged traces. After the aggregation is completed, the KPI value of the Petri net finally extracted becomes the KPI threshold.

5 Experiments

This section describes the purpose and overview of the experiment, the tools and data sets used in the experiment, and presents the results of the experiment. Finally, a discussion of the experimental results is presented.

To evaluate the effectiveness of the proposed method, an experiment was conducted to compare the proposed method with the method of abe et al. [13]. The purpose of the experiment is to compare the KPI thresholds, process models, and threshold detection times of the existing and proposed methods and to evaluate their effectiveness.

In the experiment with synthetic event logs, we apply the method to two synthetic logs with different number of variants and average trace size and compare the results. Experiments with real-life event logs are conducted by applying the method on event logs output from real business processes and comparing the results. All experiments are performed on a desktop PC with an Intel Core i7-7700 CPU 3.60GHz, 16GB memory, and Windows 10.

5.1 Tools and datasets

In Experiment 1, we use synthetic event logs generated by PLG2 (Processes Logs Generator) [17], a log generation tool using probabilistic context-free grammars. PLG2 is a tool that can generate an event log by setting arbitrary activity start and end times, resources, and other information to a process model.

In Experiment 2, we use BPI Challenge 2020 , which is a real-life event log collected from Eindhoven University of Technology (TU/e). BPI Challenge 2020 is an event log of the actual travel expense payment request process executed at TU/e, with process instances recorded for a total of two years, from 2017 to 2018.

The proposed method in this study uses the plugin "Calculate Graph Edit Distance Similarity" implemented in ProM (Process Mining Framework) to calculate the similarity as described in section 4.1.4. ProM is an open-source software for process mining, and various algorithms including process discovery are provided as plug-ins. In addition, "Calculate Graph Edit Distance Similarity" implements Greedy, A*, and Process Heuristics as graph matching algorithms. The Greedy algorithm was chosen as the default algorithm for the plugin because it is more suitable for similarity calculations based on the results of a study conducted by [16], which measured the accuracy and computation time of graph matching algorithms. The initial parameters wskipn, wskipe, and wsubn for the weights of the editing operations were all set to 1, which is the default value.

5.2 Experiment 1

In this section, we apply the existing and proposed methods to two synthetic event logs with different number of variants and average trace size and compare the threshold detection time. The threshold detection time is calculated as the sum of the Petri net similarity computation times.

The synthetic log was created by simulating 100 traces from a process model consisting of only XOR split/join. In addition, an event log was also created by simulating 100 traces from a process model in which only one XOR split was replaced by an AND split/join. AND branching executes all the sequences to be transitioned in parallel. Therefore, a process consisting of only XOR branches will execute fewer events, resulting in a smaller number of variants and a smaller average trace size. Conversely, a process that includes AND branches in addition to XOR branches increases the number

of events executed, resulting in a larger number of variants and larger average trace size. By taking advantage of these characteristics and including AND branches in only one of the models, it is possible to generate two logs with extremely different numbers of variants and average trace sizes.

The number of cases, number of events, number of variants, and average trace size of the two synthetic logs generated by PLG2 [17] are shown in Table 2. To compare the threshold detection times under the same conditions, no similarity threshold was set in the experiment using the synthetic logs, and the aggregation was repeated until the KPI threshold was two regions, i.e., two Petri nets, and the total similarity computation time was measured.

Table 2. Number of cases, events, variants, and average trace size in the synthetic event log

synthetic event log	cases	events	variants	average trace size
XOR log	100	548	12	5.48
XOR and AND log	100	1384	69	13.84

Figure 3 shows the results of a comparison of the threshold detection times for event logs containing only XOR branches and logs containing AND branches using the existing method. The comparison results show that the detection time for both cases is positively proportional to k, so it can be inferred that the detection time increases for k > 15 is also expected to increase. In the case of a log consisting only of XOR branches, the detection time for k = 15 is about 17.5 seconds, but in the case of a log including AND branches, the detection time for k = 15 is about 1,200 seconds, or about 20 minutes. Considering that the detection time increases even for k > 15, it is difficult to determine the initial number of groups k by trial and error for logs with large average trace length and number of variants, such as logs including AND branches.

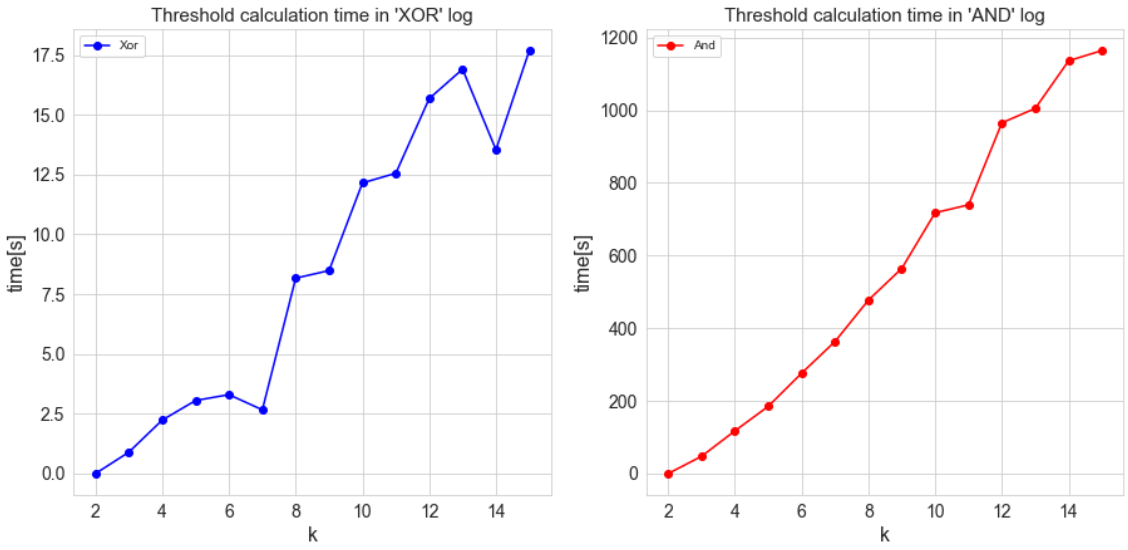


Figure 3. Threshold detection time with existing method [13] (left: logs with XOR branches only, right: logs with AND branches)

Next, the threshold detection times of the proposed and existing methods are compared, and the results are shown in Table 3 and 4. Since the number of groups of traces automatically segmented by the proposed method was 12 and 89, respectively, the initial number of groups k of the existing method was also set to 12 and 89, respectively. Table 3 shows that the difference in threshold detection time

between the existing and proposed methods for XOR branch-only logs is about 5 seconds. On the other hand, Table 4 shows that the difference of threshold detection time between the existing and proposed methods for logs including AND branches is 1975 seconds, or about 33 minutes, indicating that the proposed method significantly reduces the threshold detection time.

Table 3. Threshold detection time for XOR branch-only logs in existing and proposed methods

	existing method	proposed method
Number of initial groups (k)	12	12
Number of final groups	2	2
threshold detection time[s]	15.7	20.6

Table 4. Threshold detection time in logs with AND branches in existing and proposed methods

	existing method	proposed method
Number of initial groups (k)	69	69
Number of final groups	2	2
threshold detection time[s]	5667.0	3692.0

Finally, the KPI thresholds and Petri nets detected in each event log are shown in Figure 4, Figure 5, Figure 6, and Figure 7. The lower left corner of each figure shows the degree of similarity between the final extracted Petri nets; the closer the number is to 0, the lower the similarity between the two Petri nets, and the closer to 1, the higher the similarity. From Figures 4 and 5, the similarity between the Petri nets of the existing method is 0.628, while that of the proposed method is 0.413. This is because the traces <A,D,V,W,X,Y,E,B> exist in two Petri nets in the existing method, while the traces <A,D,V,W,X,Y,E,B> exist in only one Petri net (P2) in the proposed method. Therefore, it is considered that there is a difference in the degree of similarity between the two. Therefore, the proposed method avoids the appearance of common traces in both Petri nets, and it is easy to visually confirm the difference in process structure by extracting process models with lower similarity.

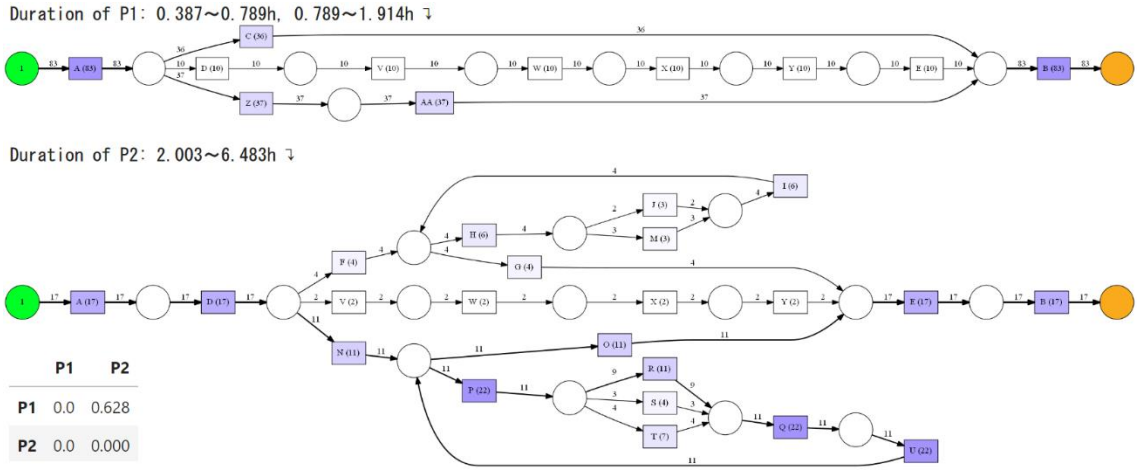


Figure 4. KPI thresholds and Petri nets for the existing method in XOR branch-only logs

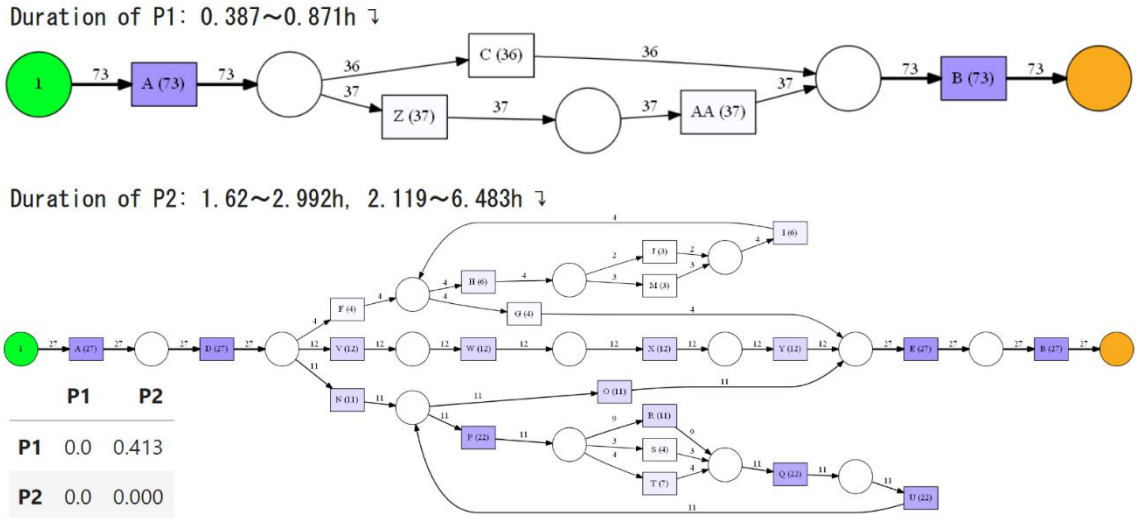


Figure 5. KPI thresholds and Petri nets for the proposed method in XOR branch-only logs

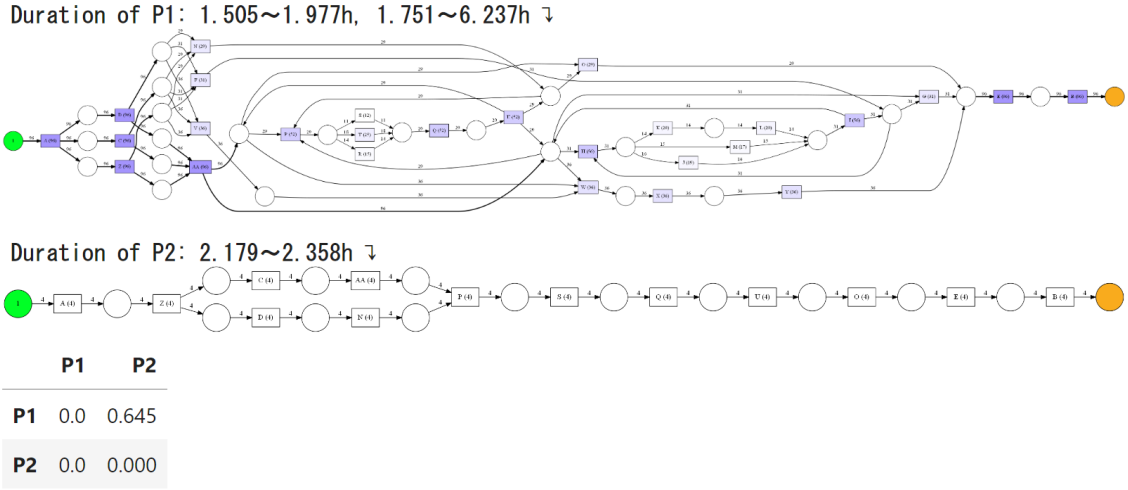
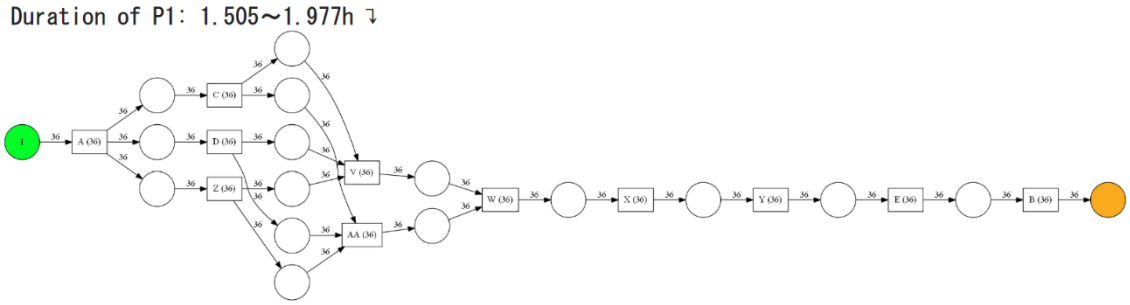
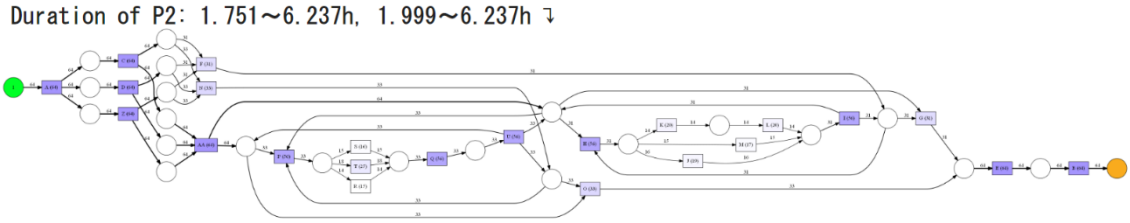


Figure 6. KPI thresholds and Petri nets for the existing method in logs containing AND

branches





	P1	P2
P1	0.0	0.636
P2	0.0	0.000

Figure 7. KPI thresholds and Petri nets for the proposed method in logs containing AND branches

It can also be confirmed from each figure that the Petri net of the log containing AND branches is more complex than the Petri net of the log containing only XOR branches. This is due to the parallel execution of events in the model including AND branches, which can be seen from the Table 2, and the larger number of events, variants, and average trace size in the logs, which can be inferred to have increased the complexity of the model. Therefore, the reason for the longer threshold detection time for the logs containing AND branches in Figure 3 and Table 4 is thought to be due to the increased complexity of the Petri net for the logs containing AND branches.

5.3 Experiment 2

In this section, we compare the results of KPI threshold detection and process discovery by the proposed and existing methods using the actual event log, BPI Challenge 2020. The number of cases, events, variants, and average trace size for BPI Challenge 2020 are shown in table 5. In our preliminary experiments, we set the similarity threshold for terminating iterations between the existing and proposed methods to 0.7. This is because in the preliminary experiment with a similarity threshold of 0.6, the existing method iterated aggregation until there were two Petri nets. Since this was judged to be too much aggregation, 0.7 was set. The process discovery algorithm was based on inductive mining [14].

Table 5. Number of cases, events, variants, and average trace size for BPI Challenge 2020

Number of cases	6886
Number of events	36796
Number of variants	89
average trace size	5.344

Table 6 shows the results of the comparison of threshold detection times for the existing and proposed methods in BPI Challenge 2020. First, we ran the existing method with an initial group size of 10 and tallied the results until we finally had four Petri nets. In the proposed method, the number of initial groups is 89 because the number of variants in the trace is 89, and the number of Petri nets is aggregated to 14 groups in the end. As a result, the threshold detection time with the existing method was about 90 minutes, while with the proposed method it was about 15 minutes.

Table 6. Comparison of threshold detection times for existing and proposed methods

	existing method	proposed method
Number of initial groups (k)	10	89
Number of final groups	4	14
threshold detection time[min]	89.13	15.15

Next, the KPI thresholds and Petri nets detected by the existing and proposed methods are shown in Figure 8 and Figure 9. From each figure, it can be confirmed that the Petri nets extracted by the existing method are more complex than those by the proposed method. This indicates that the process models by the existing method are difficult to compare, while those by the proposed method are easy to compare. For example, the proposed method shows that there are 72 "SAVED by EMPLOYEE" processes with a "duration" of 0 set as a KPI, but using the existing method, the process model is too complex to know which model "SAVED by EMPLOYEE" exists.

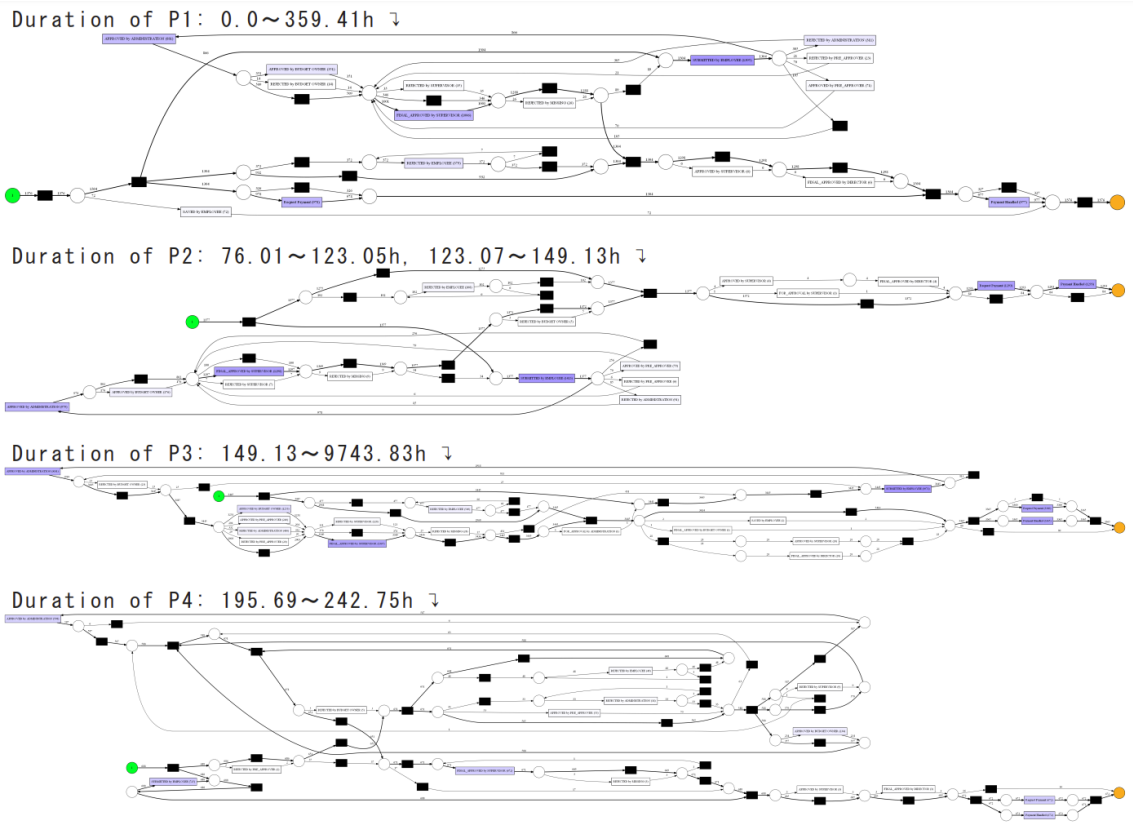


Figure 8. KPI thresholds and Petri nets detected by the existing method

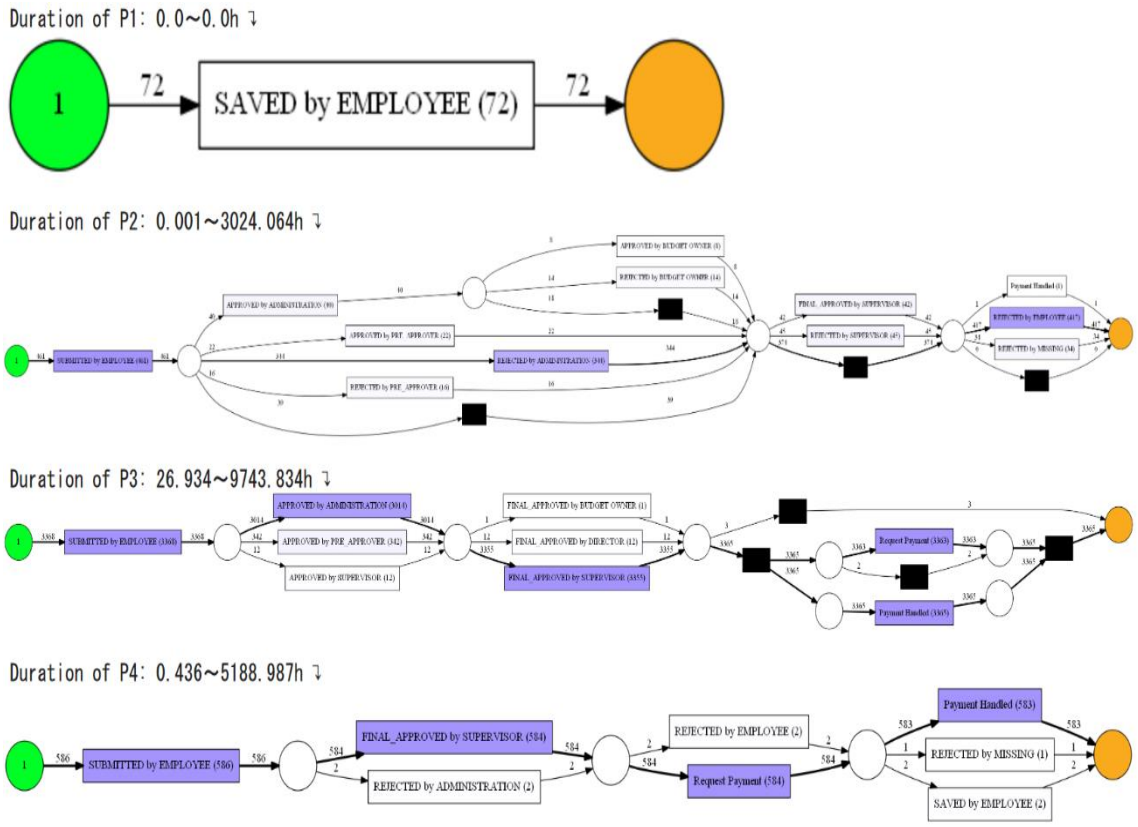


Figure 9. KPI thresholds and Petri nets detected by the proposed method

5.4 Discussion

In this section, we discuss the results of two experiments. Experiments using synthetic event logs showed that the proposed method reduced the threshold detection time for logs with a relatively large number of variants and average trace size. This may be because the proposed method divides the logs by trace variant and groups them so that multiple variants are not mixed, thereby reducing the similarity calculation time. On the other hand, if the logs were simply sorted by KPI and divided into k equal groups as in the existing method, the similarity calculation time would have increased because various variants would have been mixed in each group.

Experiments using real-life event logs similarly showed that the proposed method reduced the threshold detection time, and furthermore, the extracted Petri nets were simplified and easy to compare. The reason why the process model extracted by the existing method [13] is more complex than the one extracted by the proposed method may be because the number of initial groups k is smaller than the number of cases in the event log. In the experiment, the number of initial groups of the existing method is 10, and the number of cases in BPI Challenge 2020 is 6886, so the number of traces for each group is 6886/10 ≒ 688.

In contrast, since the initial number of groups in the proposed method is 89, the average number of traces for each group is 6886/89 ≒ 77. Therefore, the number of traces for each group of the existing method is larger than that of the proposed method, which is considered to complicate the model.

This problem can be solved by further increasing the number of initial groups in the existing method. However, the existing method takes about 90 minutes to detect the threshold when the number of

initial groups k is 10, and considering that the number of initial groups k is proportional to the computation time, it is laborious to perform trial-and-error while increasing the value of k. On the other hand, the proposed method requires only about 15 minutes to detect the threshold value even though the number of initial groups is 89, so the proposed method saves time in analysis rather than trial and error for the number of initial groups k using existing methods.

The above shows that the proposed method is more effective than the existing methods in terms of the structure of the process model and the threshold detection time. Since the proposed method automatically detects the number of initial groups k as the number of trace variants, the value of k may become huge when applied to larger logs. When the value of k becomes huge, the number of combinations of process models to compare similarities increases explosively, and the threshold detection time increases accordingly.

Therefore, if the value of k is expected to be huge, it is necessary to set k as an arbitrary value using existing methods or to define an upper limit of k in advance and execute the proposed method.

Finally, based on the KPI threshold detection results of the proposed method using real-life event logs, we provide a discussion on the BPI Challenge 2020 process. First, as described in section 5.3, there are 72 processes 'SAVED by EMPLOYEE' whose 'Duration' is 0, and it can be confirmed from Figure 9 that they are independent as a single Petri net. Therefore, Duration = 0 is the threshold value. The fourth Petri net from the top (P4) contains 583 traces where the travel expense claim was approved and 2 traces where the travel expense claim was denied, respectively. This is because the size of the two traces is 4 and they are aggregated into one Petri net, which is highly similar. However, since the KPI threshold for P4 ranges from 0.436 to 5188.987 hours, it can be inferred that the 583 traces for which travel expense claims were approved do not have a constant execution time. The Petri nets in Figure 9 are part of the 14 Petri nets finally extracted, but after checking the 14 Petri nets and each KPI thresholds, we found no correlation between 'Duration' and the Petri net pattern. Generally, the size of the process model decreases with shorter 'Duration' and increases with longer 'Duration', but no relationship was found between 'Duration' and Petri nets for the BPI Challenge 2020 process.

Thus, we found no correlation between 'Duration', which is the execution time of the process, and the process pattern. However, to obtain more useful information from KPI thresholds, it is necessary to detect and examine KPI thresholds other than 'Duration'. For example, since the event log of BPI Challenge 2020 records the resources that executed tasks, we can analyze the relationship between resources and process patterns by applying the proposed method using resources as KPIs.

6 Conclusions

In this paper, we proposed an automatic KPI threshold detection method based on trace variants, which is an improvement of the KPI threshold calculation method [13] by Abe et al. In the proposed method, event logs are first sorted based on trace variants, and the logs are divided by variant. Next, a process discovery algorithm is used to generate Petri nets from each of the partitioned traces, and a similarity table between Petri nets is calculated based on the graph edit distance. The pairs of Petri nets with the highest similarity are then aggregated into one and the similarity table is updated. By repeating this process until all similarities are below a certain level, we can extract KPI thresholds that visualize the differences in process models.

Experimental results show that the processes detected by the proposed method have a simplified structure and can be easily compared among threshold values, while the processes detected by the

existing method have a complex structure and are difficult to compare among threshold values. Furthermore, it was confirmed that the threshold detection time by the proposed method is much shorter than that by existing methods. This is thought to be because the proposed method simplifies the Petri nets in each group by grouping the variants so that they are not mixed, thereby reducing the computation of the graph edit distance between Petri nets.

In conclusion, the KPI threshold detection method based on trace variants is more effective than existing methods in terms of process model structure and threshold detection time, and further reduces the time required for business process analysis.

There are two main issues to be addressed in the future. First, the proposed method needs to be further accelerated to detect KPI thresholds in larger real-life event logs. In this paper, it was shown that the similarity computation time increases with the average trace length and the number of variants in the event log, and the threshold detection took about 15 minutes even for the size of the real-life event log used in the experiments. Therefore, if the average trace length and number of variants are even larger than the real-life event log used in this study, the threshold detection time is expected to increase significantly. This can be reduced by choosing a faster graph matching algorithm, but the accuracy of the similarity must be compromised to some extent due to the NP-complete graph edit distance problem.

References

[1] Van der Aalst, W.M. Process mining: data science in action; Springer, 2016.

[2] Grossmann, W.; Rinderle-Ma, S. Fundamentals of business intelligence; Springer, 2015.

[3] Calabrò, A.; Lonetti, F.; Marchetti, E. KPI evaluation of the business process execution through event monitoring activity. In Proceedings of the 2015 International Conference on Enterprise Systems (ES). IEEE, 2015, pp. 169–176.

[4] Dogan, O. A process-centric performance management in a call center. Applied Intelligence 2022, pp. 1–14.

[5] Wetzstein, B.; Leitner, P.; Rosenberg, F.; Brandic, I.; Dustdar, S.; Leymann, F. Monitoring and analyzing influential factors of business process performance. In Proceedings of the 2009 IEEE International Enterprise Distributed Object Computing Conference. IEEE, 2009, pp. 141–150.

[6] Ghasemi, M.; Amyot, D. Goal-oriented process enhancement and discovery. In Proceedings of the Business Process Management: 17th International Conference, BPM 2019, Vienna, Austria, September 1–6, 2019, Proceedings 17. Springer, 2019, pp. 102–118.

[7] Takei, T.; Horita, H. Towards Goal-Oriented Business Process Model Repair. In Proceedings of the 2021 10th International Congress on Advanced Applied Informatics (IIAI-AAI). IEEE, 2021, pp. 691–696.

[8] Horita, H.; Hirayama, H.; Tahara, Y.; Ohsuga, A. Towards Goal-Oriented Conformance Checking. In Proceedings of the SEKE 2015, pp. 722–724.

[9] Da˛browski, J.; Kifetew, F.M.; Muñante, D.; Letier, E.; Siena, A.; Susi, A. Discovering requirements through goal-driven process mining. In Proceedings of the 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW). IEEE, 2017, pp. 199–203.

[10] Ponnalagu, K.; Ghose, A.; Narendra, N.C.; Dam, H.K. Goal-aligned categorization of instance variants in knowledge-intensive processes. In Proceedings of the Business Process Management: 13th International Conference, BPM 2015, Innsbruck, Austria, August 31–September 3, 2015, Proceedings 13. Springer, 2015, pp. 350–364.

[11] Yan, J.; Hu, D.; Liao, S.S.; Wang, H. Mining agents' goals in agent-oriented business processes. ACM Transactions on Management Information Systems (TMIS) 2014, 5, 1–22.

[12] Guizzardi, R.; Reis, A.N. A method to align goals and business processes. In Proceedings of the Conceptual Modeling: 34th International Conference, ER 2015, Stockholm, Sweden, October 19-22, 2015, Proceedings 34. Springer, 2015, pp. 79–93.

[13] Abe, M.; Kudo, M. Analyzing business processes by automatically detecting kpi thresholds. In Proceedings of the 2016 IEEE International Conference on Services Computing (SCC). IEEE, 2016, pp.

187–194.

[14] Leemans, S.J.; Fahland, D.; Van Der Aalst,W.M. Discovering block-structured process models from event logs-a constructive approach. In Proceedings of the Application and Theory of Petri Nets and Concurrency: 34th International Conference, PETRI NETS 2013, Milan, Italy, June 24-28, 2013. Proceedings 34. Springer, 2013, pp. 311–329.

[15] Verbeek, H.; Buijs, J.C.; Van Dongen, B.F.; Van Der Aalst,W.M.; et al. XES, XESame, and ProM 6. In Proceedings of the CAiSE Forum (Selected Papers). Springer, 2010, Vol. 72, pp. 60–75.

[16] Dijkman, R.; Dumas, M.; García-Bañuelos, L. Graph matching algorithms for business process model similarity search. In Proceedings of the International conference on business process management. Springer, 2009, pp. 48–63.

[17] Burattin, A. PLG2: multiperspective processes randomization and simulation for online and offline settings. arXiv preprint arXiv:1506.08415 2015.