International Journal of Service and Knowledge Management
International Institute of Applied Informatics
2023, Vol. 7, No. 1, IJSKM691

Comparison of Goal-Oriented Business Process Model Repair and Discovery

Taro Takei *, Hiroki Horita *

Abstract

Process mining is a technology that extracts useful knowledge from event logs output from information systems and utilizes it for business process analysis and improvement. Conventional process mining approaches do not take into account information about organizational goals, and thus may output process models that inhibit the goal of the organization, for example, "to reduce cost". To solve such problems, goal-oriented process discovery method was proposed as an existing research. On the other hand, one of the methods to generate process models is process model repair. This method uses a process model and an event log as input, and repairs the existing process model to reproduce all the behaviors observed in the event log. However, the existing process model repair methods do not take into account the information about goals. Therefore, in this paper, we propose a goal-oriented process model repair method. It is easy to visually identify and compare which parts of the existing process model satisfy the goal satisfaction and which parts do not satisfy the goal. In addition, the structure of the model is simpler and closer to the initial model, and its effectiveness can be demonstrated. The effectiveness of the model was confirmed by experimental results.

Keywords: process mining, goal, business process, business process modeling

1 Introduction

Log data, which is a record of daily business activities, can be easily obtained from information systems. The analysis of such log data is called process mining [1], and it is becoming more and more important to improve existing business processes by analyzing the data obtained in business processes. In order to improve business processes, it is first necessary to analyze the current state and understand how business processes are being executed. By using the process discovery method [2], we can automatically construct a business process model that aggregates the behavior of business processes from event logs that are executed in reality and record their contents, which can be useful for as-is analysis.

Methods for as-is analysis are often activity-oriented and do not take into account the specific goals pursued by individual cases in the business process or the goals pursued by

* Ibaraki University, Ibaraki, Japan

stakeholders [3], [4]. However, since companies have goals such as ”cost reduction” and ”improvement of customer satisfaction” that are stated by the organization and its stakeholders, it is desirable to conduct analyses that take these characteristics into account. In order to deal with such problems, Ghasemi et al. proposed a goal-oriented process discovery method [3]. In this method, three criteria for goal satisfaction are defined, and process discovery is performed based on each criterion to automatically generate a business process model that considers the satisfaction level of the goal.

Apart from process discovery, one of the techniques for generating new business process models is process model repair [5]. This is a technique to repair a process model so that it can reproduce all the behaviors in the event log by inputting the event log and an existing process model and calculating the alignment that represents the degree of divergence between the two. The repaired process model can be compared with the existing process model, and we can visually see how the process has changed from the past. However, as well as the problem mentioned earlier, conventional process model repair methods do not take into account the information about the goal, so they may output a process model that ignores the goal.

In this paper, we propose a goal-oriented process model repair method that combines the goal-oriented process discovery method proposed by Ghasemi et al [3] and the process model repair method proposed by Fahland et al [5] in order to incorporate information about goals into the process model repair method. In this method, we first construct an enhanced event log which is an event log enhanced by the degree of goal satisfaction, using a program devised by Ghasemi et al. in [6] as a preprocessing of data. Next, we adopt one of the three criteria for goal satisfaction proposed by the same authors, and extract traces that satisfy the criteria from the enhanced event log. Finally, we perform Fahland's process model repair using the extracted traces and the existing process models as input to realize the goal-oriented process model repair method. By generating a goal-aware process model and comparing it with the existing process model, we can identify the parts of the existing model that do not satisfy the goal and the parts that do satisfy the goal. In addition, unlike existing methods, this method generates process models by repairing process models, so the structure of the repaired models is similar to that of the input models, making comparison easier.

The paper is structured as follows. In section 2, we introduce preliminaries about this paper. In section 3, we introduce related works that we have referred to in realizing our method. In section 4, we describe the proposed method. In section 5, we present and discuss the experimental results of comparing the proposed method with the existing process models. Finally, in section 6, we summarize this paper and discuss future works.

2 Preliminaries

We describe the petri nets, event logs, and process model repair used in this paper.

2.1 Petrinets

Business process modeling is a means of graphically and formally representing business processes. Petri nets are a type of business process modeling language that can express various relationships related to the execution of events, such as XOR and AND. The following is a definition of Petri nets.

Definition 1. *(Petri net) A Petri net is a triplet N = (P,T,F) where P is a finite set of places, T is a finite set of transitions such that* $P \cap T = \emptyset$*, and* $F \subseteq (P \times T) \cup (T \times P)$ *is a set of directed arcs, called the flow relation. A marked Petri net is a pair (N,M), where N = (P,T,F) is a Petri net and where* $M \in \mathbb{B}(P)$ *is a multi-set over P denoting the marking of the net. The set of all marked Petri nets is denoted N [1].*

Fig. 1 shows a simple process model described by a Petrinet. This petrinet starts at place p1 on the left and ends at place p6 on the right. Rectangles are transitions, which are also called events.

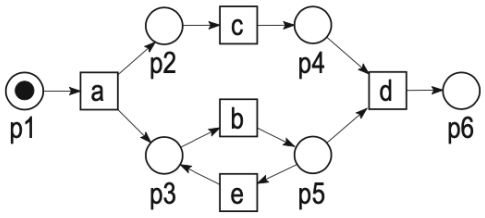

Figure 1: A simple petrinet [5]

2.2 Event Logs

Event Log is recorded by information systems as a result of events executed in business processes. The following are the definitions of events, traces, and logs that make up the event log.

Definition 2 . *(Event, Trace, Log) Let* $\mathbb{E}$ *be the event universe, i.e., the set of all possible event identifiers. Events may be characterized by various attributes, e.g., an event may have a timestamp, correspond to an activity, is executed by a particular person, has associated costs, etc. Let AN be a set of attribute names. For any event* $e \in \varepsilon$ *and name n* $\in$ *AN,# n(e)is the value of attribute n for event e. If event e does not have an attribute named n, then # n(e) =* $\perp$ *(null value) [1].*

2.3 Process Model Repair

Fahland et al. proposed a method for repairing a process model so that it can reproduce all the behaviors recorded in the event log [5]. This paper is the first to focus on process model repair in process mining. Conventional process mining approaches have been able to find the differences between the behaviors observed in the process models and the behaviors observed in the event logs by using conformance checking [7], but they have not supported a method to repair the process models based on the differences [5]. In addition, process discovery is a technique for generating new process models, but the process models obtained by process discovery are likely to have no similarity to the existing process models in the organization, and the newly discovered process models may lose the value of the original models because they do not consider the structure of the existing models [5]. Therefore, Fahland et al. proposed a method to repair the existing model so that it can reproduce all the behaviors observed in the event log while maintaining the structure of the existing process model as much as possible. The procedure of the repair method of Fahland et al is as follows [5].

1. Given an event log L and a model N, determine the multiset L_f of fitting traces and the multiset L_n of non-fitting traces.

2. Split the multiset of non-fitting traces L_n into L_d and L_u. Traces L_d are considered as deviating and the model needs to be repaired to address these. Traces in L_u could be considered as outliers/noise and do not trigger repair actions.

3. Repair should be based on the multiset $L' = L_f \cup L_d$ of traces. L' should perfectly fit the repaired model N', but there may be many candidate models N'.

4. Return a repaired model N' that can be easily related back to the original model N, and in which changed parts are structurally simple.

In this paper, we incorporated the idea of goal orientation into this paper.

3 Related Works

In this section, we describes related works. In section 3.1, we describe goal-oriented business process design and validation. In section 3.2, we describe goal-oriented process mining. In section 3.3, we desctibe about process model repair.

3.1 Goal-Oriented Business Process Model Design and Validation

Goal models used in goal-oriented requirements analysis can be used to design and verify business process models. Horita et al [8], sun et al [9], Nagel et al [10] proposed a method to transform goal models into business process models. By doing so, we can clarify what kind of business process is required to achieve the goal in the organization. Nagel et al [11] and Gröner et al [12] proposed a method for verifying the consistency of goal models and business process models. By doing so, it is possible to detect inconsistencies between models and to detect flows that do not achieve the goal. All of these methods are used in the design of business processes. Therefore, they do not take into account how the business process is actually executed. On the other hand, data-driven methods based on process mining, such as our research, can be used for business process design by considering the actual state of business process execution.

3.2 Goal-Oriented Process Mining

Although much research on goal orientation has been done in the field of requirements engineering [13] [14], it has not been done much in the field of process mining. Process mining methods that use goal orientation include goal-oriented conformance checking [15], goal repair [16], agent goal learning [17], aligning goal and business process[18], [19], automatic determination of KPI threshold values[20], discovering requirements[21] and variant analysis [22].

In the following, we explain goal-oriented process discovery, which is most relevant to this paper. Ghasemi et al. proposed Goal-Oriented Process Discovery (GoPED), a process discovery method that considers the degree of goal satisfaction [3]. GoPED refers to KPI (Key Performance Indicators) as information about goals. KPIs are indicators to quantitatively evaluate the performance and sufficiency of an organization's goals (e.g., processing time of a process).

In GoPED, traces are filtered based on three criteria (case perspective, goal perspective, and organization perspective), and traces with high goal satisfaction are selected. By using the traces and performing process discovery using the α-algorithm [2], we can obtain a process model that takes the goal into account. In this paper, we modified the process discovery step of GoPED into a process model repair method to realize a goal-oriented process model repair method. By doing so, we can expect that the repaired process model is close to the existing process models and it is easy to grasp the influence of goal satisfaction.

3.3 Process Model Repair

Dees et al [23] proposed a method to repair a process model to achieve a better KPI level using classification tree learning and Fahland et al's process model repair [5] described in the section 2.3.

As problems of this method, Dees et al. mentioned that it is time-consuming and prone to human errors because all the steps are executed manually, and that it only deals with time-related KPIs. On the other hand, the proposed method of this paper, Goal-Oriented Process Model Repair, requires fewer steps to repair a process model than Dees et al.'s method, and can also deal with KPIs related to cost.

4 Proposed Method

In this section, we explain a goal-oriented business process model repair method that integrates a goal-oriented process discovery method [3] and a business process model repair method [5]. The flow of the proposed method consists of 1: Calculation of goal satisfaction and enhancing event logs (section 4.1), 2: Filtering of event log based on goal satisfaction (section 4.2), and 3: Business process model repair (section 4.3). 1 and 2 are based on Ghasemi et al. [6], [3], and 3 is based on Fahland et al. [5]. An overview of the proposed method is shown in Fig. 2.

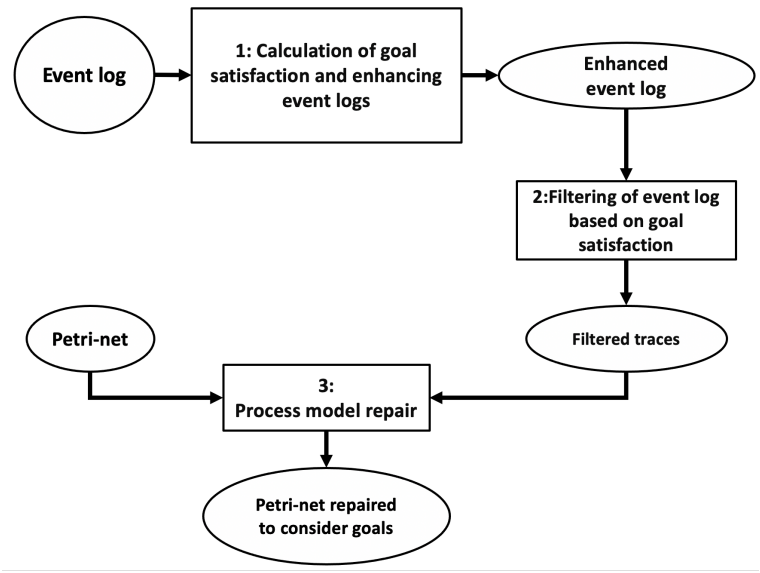

Figure 2: Overview of our proposed method

The first input event log is an event log consisting of case identifier, activity, timestamp, and cost. Examples of the input event log and the enhanced event log to be generated are shown in Table 1 and Table 2. Table 1 will be converted to Table 2 through the process described in Section 4.1.

Table 1: Initial event logs

Case ID	Activity	Timestamp	Cost
1	A	1970-01-01 09:00:00+09:00	100
1	B	1970-01-01 09:26:22+09:00	200
1	D	1970-01-01 09:42:46+09:00	150
1	F	1970-01-01 10:07:41+09:00	100
1	G	1970-01-01 10:31:56+09:00	300
1	H	1970-01-01 11:00:53+09:00	0
1	K	1970-01-01 11:29:08+09:00	100
2	A	1970-01-01 09:00:00+09:00	100
2	B	1970-01-01 09:21:10+09:00	200
2	F	1970-01-01 09:40:12+09:00	100
⋮	⋮	⋮	⋮

Table 2: enhanced event logs

Case ID	Trace	G1	G2
1	A,B,D,F,G,H,K	90	100
2	A,B,F	87	100
3	A,B,D	85	100
4	A,B,D	85	100
5	A,B,F,G	77	75
6	A,B,F,G	70	75
7	A,B,F,G	65	75
8	A,B,D,G,I	63	65
9	A,B,D,G,I	63	65
10	A,B,D,G,I	59	65
⋮	⋮	⋮	⋮

4.1 Calculation of Goal Satisfaction Enhancing Event Logs

In order to calculate the goal satisfaction, we first set KPIs and extract the corresponding points from the event log. In this paper, we set ”process execution time” and ”process execution cost” in each case as KPIs related to Goal 1 ”reduction of process execution time” and Goal 2 ”reduction of process cost”. The ”process execution time” is calculated as ”(time of last activity) - (time of first activity)” in a trace. The ”process execution cost” is the sum of the costs of each activity in a trace. These are obtained for each trace.

Next, we calculate the satisfaction of the goal using EnhancedLogMaker [6]. Before calculating the degree of satisfaction, we first set the target, threshold, and minimum values of the degree of satisfaction as parameters. The target value is the KPI value at which the

satisfaction level is 100, the threshold value is the KPI value at which the satisfaction level is 50, and the minimum value is the KPI value at which the satisfaction level is 0.

For example, in the goal "reduction of process cost", the target, threshold, and minimum values are set to 1000, 2000, and 3000, respectively. In this case, a trace with a total cost of 1000 has a sufficiency of 100, a trace with a total cost of 2000 has a sufficiency of 50, and a trace with a total cost of 3000 has a sufficiency of 0.

The degree of fulfillment of the final goal is calculated by the following equation [6]. In the formula, the degree of sufficiency is denoted by SL, the value of KPI by Current v, the target value by Target v, the threshold value by Threshold v, and the minimum value by Worst v. These equations are used for maximization and minimization. In the case of cost, we want to minimize it, so the equation below is used.

For Maximization:

$$SL = 100 \times \begin{cases} \frac{1}{2} + \frac{|Current\ v - Threshold\ v|}{2|Target\ v - Threshold\ v|}, \\ if\ Current\ v \geq Threshold\ v \\ \frac{|Current\ v - Worst\ v|}{2|Threshold\ v - Worst\ v|}, \\ if\ Current\ v < Threshold\ v \end{cases}$$

For Minimization:

$$SL = 100 \times \begin{cases} \frac{1}{2} + \frac{|Current\ v - Threshold\ v|}{2|Target\ v - Threshold\ v|}, \\ if\ Current\ v < Threshold\ v \\ \frac{|Current\ v - Worst\ v|}{2|Threshold\ v - Worst\ v|}, \\ if\ Current\ v \geq Threshold\ v \end{cases}$$

After calculating the satisfiability of each goal in all traces, we add columns G1 and G2 as shown in Table 2, and add the satisfiability SL of each goal to make an extended event log.

4.2 Log Filtering Based on Goal Satisfaction Level

Next, we filter the "enhanced event log" by the criteria related to goal satisfaction and obtain a subset of traces that satisfy the criteria. We adopt the criterion Q_{case} (row and case perspective), which is one of the three criteria proposed in the existing work [3]. Criterion Q_{case} is defined in [3] as follows.

$$Q_{case} = \{q_j = (G_j, \bar{sl}_j) \mid G_j \in \mathbb{G} \wedge 0 \leq \bar{sl}_j \leq 100\}, \\ 0 \leq conf \leq 1$$

Q_{case} is a set of criteria q_j, where each q_j is a tuple consisting of one goal $G_j \in \mathbb{G}$ and a sufficiency threshold $\bar{sl}_j$.

For example, if we set Q_{case} = (G1, 70), (G2, 80), conf = 0.8, we look for a case where at least 80% of the traces have G1 satisfiability greater than 70 and G2 satisfiability greater than 80. Specifically, if there are 100 traces of ⟨a, b, c, d⟩, and 80 or more traces have G1 and G2 sufficiency exceeding the criterion, then the criterion conformance rate of ⟨a, b, c, d⟩ is greater than conf = 0.8, the goal is considered to be satisfied.

Thus, by obtaining an event log consisting of a subset of traces that satisfy the criterion Q_{case}, we can generate a process model that reflects only those behaviors that increase the degree of goal satisfaction by process model repair.

4.3 Process Model Repair

Finally, using the subset of traces filtered by the criterion Q_{case} and the Petri net to be repaired as input, we use the process model repair method proposed by Fahland et al [5]. Through this process, we can obtain a process model that can replay the input event log. See section 2.3 for more information on the process model repair in Fahland et al. We use [5] because it is a representative method, but we believe that other repair methods [24], [25] can be used as well.

In the existing method [3], process discovery is performed in this step by α algorithm [2] instead of process model repair. In other words, the similarity between our method and the existing methods is that these methods create an enhanced event log and filter the log by the criteria, and the difference is that the filtered event log is used for process discovery or process model repair. In the experiments in section 5, we compare the process models generated by our method and existing methods.

5 Evaluation

In this section, we compare the business process models generated by the goal-oriented process discovery method [3] and the business process models generated by the goal-oriented process model repair method proposed in this paper.

In this experiment, we automatically generate a business process model using a tool [26], and obtain an event log consisting of 1000 traces through simulation using the model. Then, we apply our method to the acquired event logs and create an enhanced event log that includes the satisfaction of Goal 1 (G1) ”reduction of process execution time”, and Goal 2 (G2) ”reduction of process cost”, as attributes. Finally, using a subset of the traces filtered by the criterion Q_{case} from the enhanced event log, we perform process discovery and process model repair using the α algorithm to generate and compare two process models. For comparison, the graph edit distance is used to measure the similarity between the original model and the model generated by each method.

In this experiment, we use PLG2 (Processes Logs Generator) [26] and ProM (Process Mining Framework)[1]. PLG2 is an event log generator [26], which can generate event logs by setting arbitrary information such as activity start/end times and resources in a process model and simulating it. ProM is an open source software for process mining, and various algorithms including process discovery are provided as plug-ins. In this study, we created Petri nets described in pnml format using PLG2, generated event logs with random timestamps and costs, and experimented with process model repair using Fahland's method [5], which is implemented as a plug-in in ProM. Similarly, the graph edit distance was calculated using the Culculate Graph Edit Distance Similarity plug-in for ProM.

Our experiments were conducted on an MacBook Air, 1.8 GHz Intel Core i5, each core being equipped with 8 GB main memory, running on macOS 10.13.6.

[1] https://www.promtools.org/doku.php

5.1 Results

The experiments were conducted four times. Based on the results of the four experiments, we present our discussion in section 5.2.

5.1.1 Results 1

The Petri net P1 used in the first experiment is shown in Fig. 3. We generated 1000 traces from P1 using PLG2, and created an enhanced event log with additional goal satisfaction. We filtered the enhanced event log by setting the selection criterion of traces as follows: G1 satisfies more than 80 percent of traces and G2 satisfies more than 80 percent of traces, i.e., Q_{case} = (G1, 80), (G2, 80), conf = 0.8. The number of traces in the event log generated by filtering was 151, and the average trace size was 21.

Fig. 4 shows the Petri net P1α generated by the process discovery using the filtered event log and the α-algorithm. Fig. 5 shows the Petri net P1Repair generated by the process model repair using the filtered event log and Petri net P1.

Finally, the graph edit distance was used to measure the similarity between the original model, P1, and each of the generated models. The resulting graph edit distance between P1 and P1α is 0.442 and between P1 and P1Repair is 0.857.

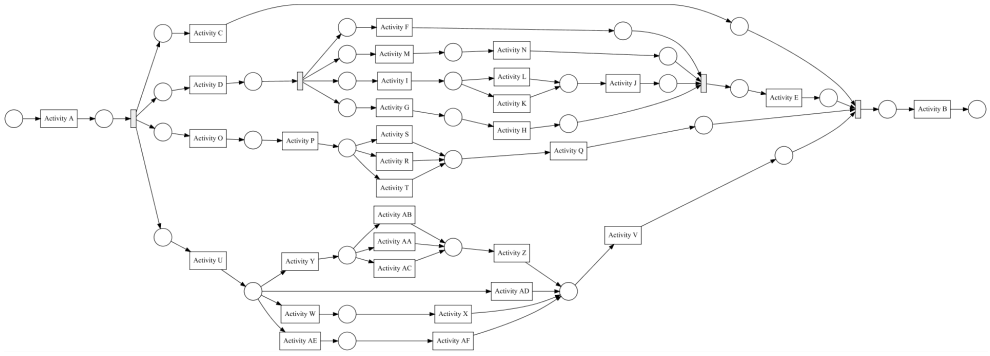

Figure 3: process model used for simulation (P1, experiment 1)

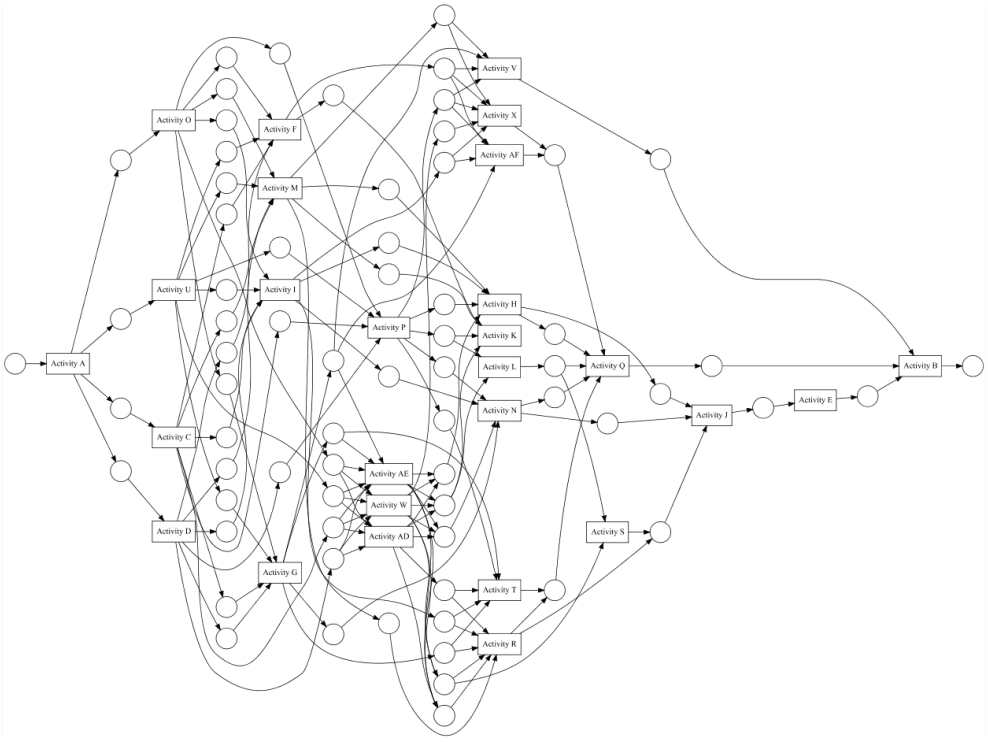

Figure 4: process model generated by process discovery (P1α, experiment 1))

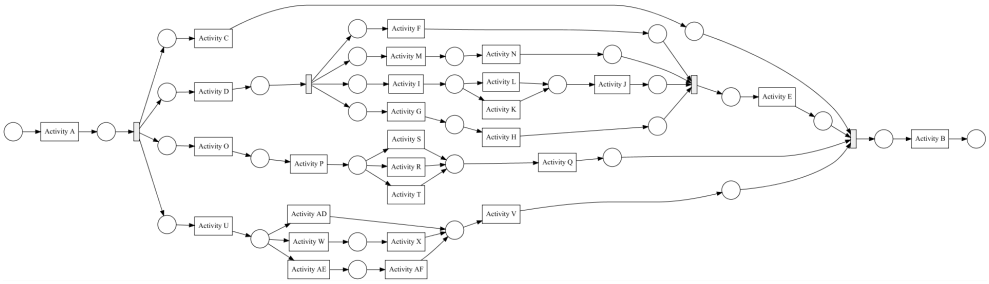

Figure 5: repaired process Model (P1Repair, experiment 1))

5.1.2 Results 2

The Petri net P2 used in the first experiment is shown in Fig. 6. We generated 1000 traces from P2 using PLG2, and created an enhanced event log with additional goal satisfaction. We filtered the enhanced event log by setting the selection criterion of traces as follows: G1 satisfies more than 70 percent of traces and G2 satisfies more than 70 percent of traces, i.e., Q_{case} = (G1, 70), (G2, 70), conf = 0.8. The number of traces in the event log generated by filtering was 373, and the average trace size was 24.

Fig. 7 shows the Petri net P2α generated by the process discovery using the filtered event log and the α-algorithm. Fig. 8 shows the Petri net P2Repair generated by the process model repair using the filtered event log and Petri net P2.

Finally, the graph edit distance was used to measure the similarity between the original model, P2, and each of the generated models. The resulting graph edit distance between P2 and P2α is 0.442 and between P2 and P2Repair is 0.846.

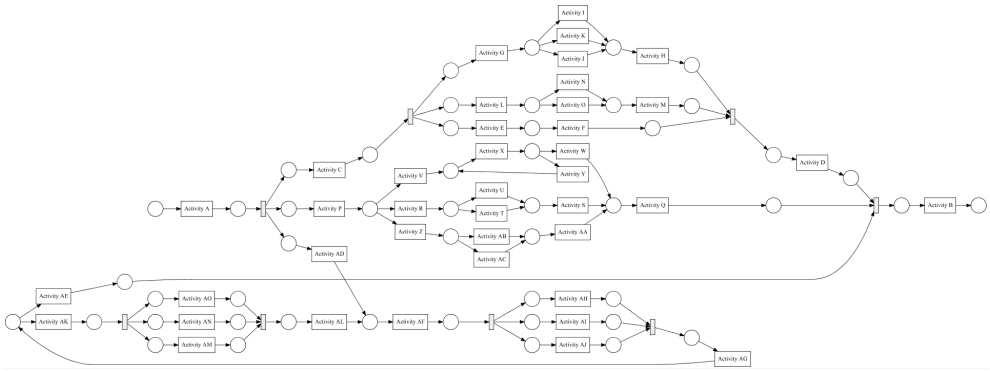

Figure 6: process model used for simulation (P2, experiment 2)

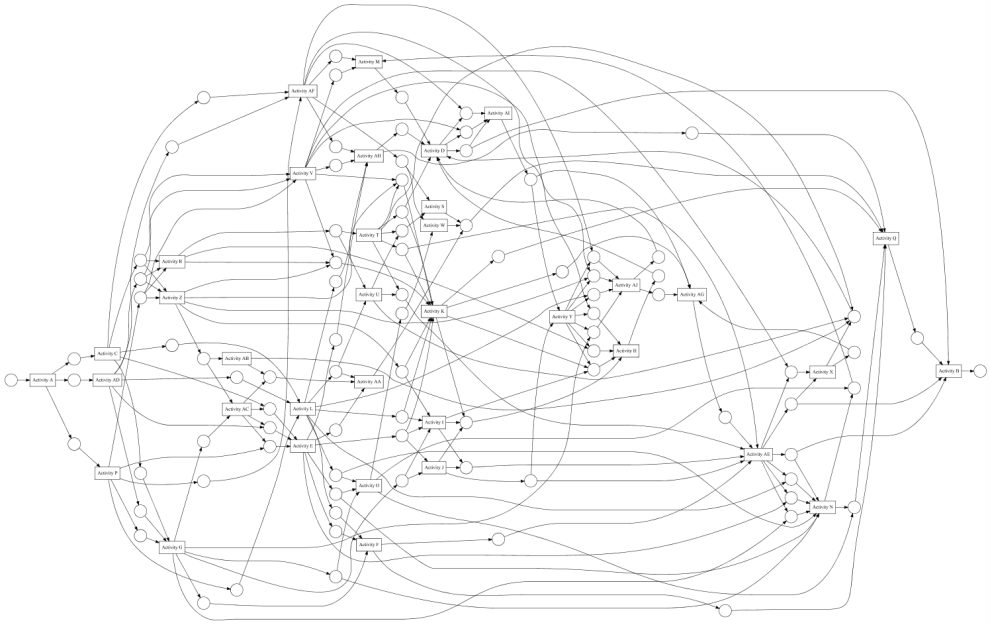

Figure 7: process model generated by process discovery (P2α, experiment 2))

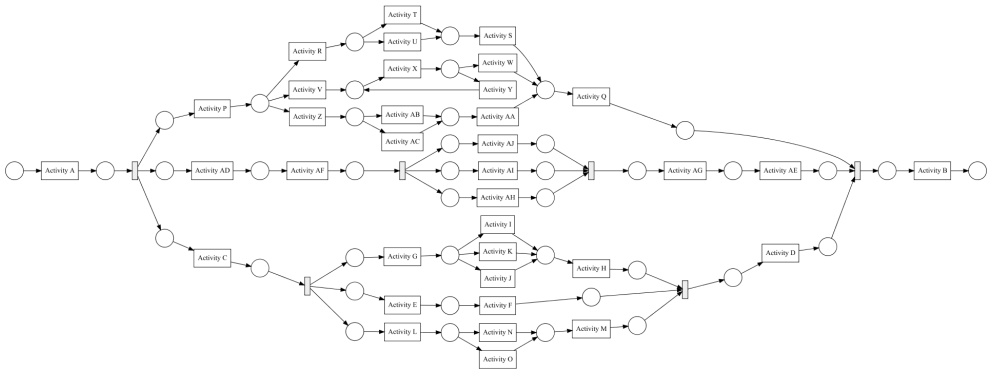

Figure 8: repaired process Model (P2Repair, experiment 2))

5.1.3 Results 3

The Petri net P3 used in the first experiment is shown in Fig. 9. We generated 1000 traces from P1 using PLG2, and created an enhanced event log with additional goal satisfaction. We filtered the enhanced event log by setting the selection criterion of traces as follows: G1 satisfies more than 70 percent of traces and G2 satisfies more than 80 percent of traces, i.e., Q_{case} = (G1, 70), (G2, 80), conf = 0.8. The number of traces in the event log generated by filtering was 481, and the average trace size was 6.

Fig. 10 shows the Petri net P3α generated by the process discovery using the filtered event log and the α-algorithm. Fig. 11 shows the Petri net P1Repair generated by the process model repair using the filtered event log and Petri net P3.

Finally, the graph edit distance was used to measure the similarity between the original model, P3, and each of the generated models. The resulting graph edit distance between P3 and P3α is 0.453 and between P3 and P3Repair is 0.764.

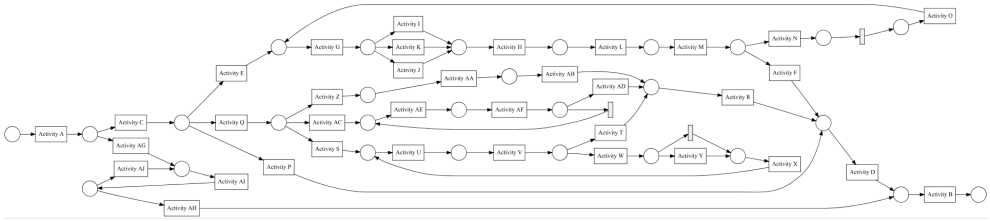

Figure 9: process model used for simulation (P3, experiment 3)

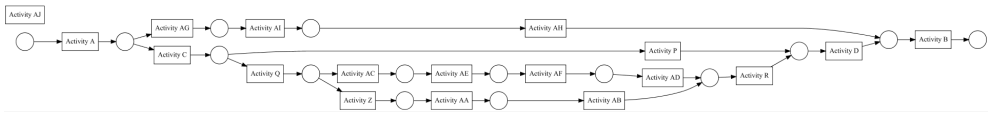

Figure 10: process model generated by process discovery (P3α, experiment 3))

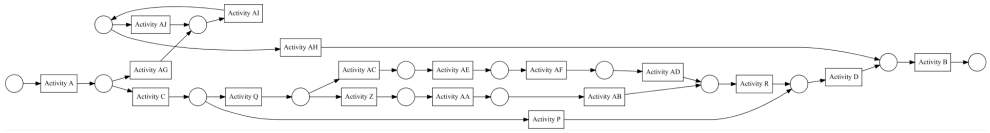

Figure 11: repaired process Model (P3Repair, experiment 3))

5.1.4 Results 4

The Petri net P4 used in the first experiment is shown in Fig. 12. We generated 1000 traces from P1 using PLG2, and created an enhanced event log with additional goal satisfaction. We filtered the enhanced event log by setting the selection criterion of traces as follows: G1 satisfies more than 80 percent of traces and G2 satisfies more than 80 percent of traces, i.e., Q_{case} = (G1, 80), (G2, 80), conf = 0.8. The number of traces in the event log generated by filtering was 383, and the average trace size was 8.

Fig. 13 shows the Petri net P4α generated by the process discovery using the filtered event log and the α-algorithm. Fig. 14 shows the Petri net P1Repair generated by the process model repair using the filtered event log and Petri net P4.

Finally, the graph edit distance was used to measure the similarity between the original model, P4, and each of the generated models. The resulting graph edit distance between P4 and P4α is 0.437 and between P4 and P4Repair is 0.705.

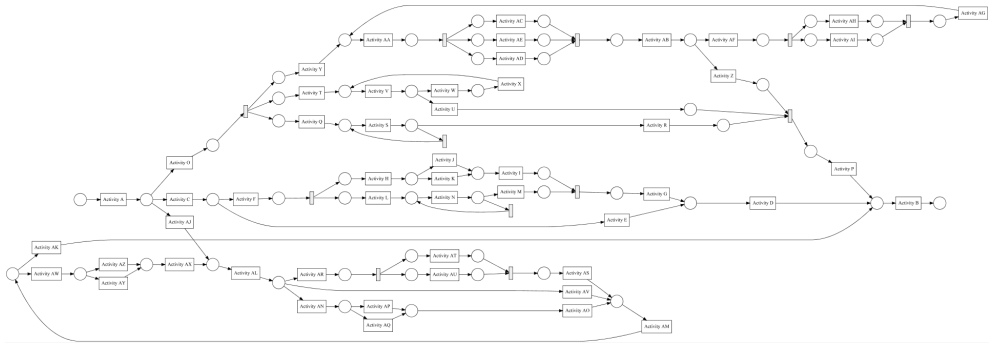

Figure 12: process model used for simulation (P4, experiment 4)

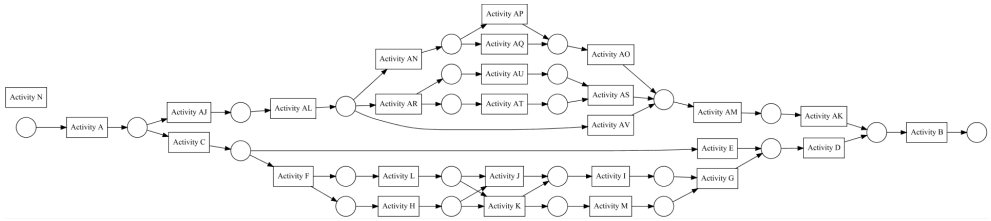

Figure 13: process model generated by process discovery (P4α, experiment 4))

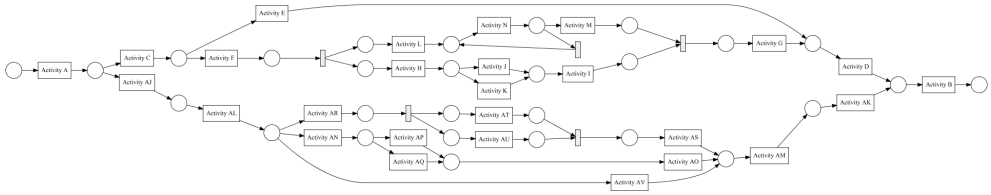

Figure 14: repaired process Model (P4Repair, experiment 4))

5.2 Discussion of the Results

First, we found that the graph edit distance of our method outperformed that of existing methods in all experiments. Thus, it is quantitatively shown that the models generated by our method are more similar to the original models than the existing methods.

In Experiments 1 and 2, the average sizes of the traces in the logs filtered by the criterion from the enhanced event log are 21 and 24, respectively, which are larger than those in Experiment 3 (size 6) and Experiment 4 (size 8). The size here is the sum of the events contained in each trace. This is probably because there are more AND-split or parallel structures in the first input Petri nets than in the Petri nets of Results 3 and 4. As a result of the increase in the average size of the traces in the log filtered from the enhanced event log, the Petri net P1α (Fig. 4) resulting from the process discovery has a large number of arcs and disordered structures. On the contrary, the Petri net P1Repair by our method (Fig.

5) has a coherent structure, a small number of arcs, and a similar structure to the input Petri net (Fig. 3). Therefore, it is easy to visually compare the input Petri net and the Petri net by our method, and to identify the parts filtered by goal satisfaction.

In Experimental Results 3 and 4, the average trace sizes of the filtered logs are 6 and 8, respectively, which are smaller than those in Experimental Results 1 and 2. Therefore, the Petri nets generated by the existing method did not have the same structure as in Experiments 1 and 2. The same results were obtained when using the business process repair method.

These results show that our method is particularly effective for business processes with large trace sizes, where there are many parallel structures.

However, one problem with this experiment is that only one simulation was performed for each experiment. To address this problem, we attempted to simulate each experiment multiple times, but found that the event logs generated by the simulations were almost similar unless the original model was changed, resulting in the same experimental results. This is because simulating 1000 traces outputs an event log that covers all the behaviors of the original model. Therefore, we believe that further experiments with different original models will enhance the effectiveness of the proposed method.

6 Conclusion

In this paper, we proposed a goal-oriented process model repair method that integrates the existing goal-oriented process discovery method [3] and process model repair method [5]. The resulting business process model guarantees the satisfaction of the defined goals, ”reduction of process time” and ”reduction of process cost,” and avoids the problems in the process model that occurred in the existing methods. These results show that our method is particularly effective for business processes with large trace sizes, where there are many parallel structures. By comparing the process models obtained in this way, it became easier to identify which parts of the existing process models satisfied the goal satisfaction and which parts did not.

As future works, the following issues are raised.

- Since the KPIs that can be added to the event log in our method are limited to ”process execution time” and ”process execution cost”, we have only considered the goals related to these two KPIs. However, since business processes have various objectives and therefore a wide range of KPIs to be considered, it is a challenge to deal with various factors including the relationships among multiple goals. There may be cases where there are inefficient but necessary processes in business processes. We may be able to discover this by considering the relationships among multiple goals.

- In addition to setting the goal items, setting the threshold is also an important issue to be considered in the future.

- It is also important to use real-life event log and goals in the evaluation.

- In addition, our method adopts the criterion Q_{case} defined in the existing method [3], but the method has two other criteria for goal satisfaction to be used in filtering the enhanced event log. Therefore, we should implement these two criteria in the future, and filter from various viewpoints.

Acknowledgments

I would like to thank the people at ibaraki university for their comments and suggestions on this research.

References

[1] Wil Van der Aalst. *Process mining data science in action*. Springer, 2016.

[2] Wil Van der Aalst, Ton Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. *IEEE transactions on knowledge and data engineering*, 16(9):1128–1142, 2004.

[3] Mahdi Ghasemi and Daniel Amyot. Goal-oriented process enhancement and discovery. In *International conference on business process management*, pages 102–118. Springer, 2019.

[4] Mahdi Ghasemi and Daniel Amyot. From event logs to goals: a systematic literature review of goal-oriented process mining. *Requirements Engineering*, 25(1):67–93, 2020.

[5] Dirk Fahland and Wil Van Der Aalst. Repairing process models to reflect reality. In *International Conference on Business Process Management*, pages 229–245. Springer, 2012.

[6] Mahdi Ghasemi and Daniel Amyot. Data preprocessing for goal-oriented process discovery. In *2019 IEEE 27th International Requirements Engineering Conference Workshops*, pages 200–206. IEEE, 2019.

[7] Wil Van der Aalst, Arya Adriansyah, and Boudewijn van Dongen. Replaying history on process models for conformance checking and performance analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(2):182–192, 2012.

[8] Hiroki Horita, Kozo Honda, Yuichi Sei, Hiroyuki Nakagawa, Yasuyuki Tahara, and Akihiko Ohsuga. Transformation approach from kaos goal models to bpmn models using refinement patterns. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, pages 1023–1024, 2014.

[9] Zhouxuan Sun, Jian Wang, Keqing He, Shujuan Xiang, and Dunhui Yu. A model transformation method in service-oriented domain modeling. In *2010 21st Australian Software Engineering Conference*, pages 107–116. IEEE, 2010.

[10] Benjamin Nagel, Christian Gerth, Jennifer Post, and Gregor Engels. Kaos4soa-extending kaos models with temporal and logical dependencies. In *CAiSE Forum*, pages 9–16, 2013.

[11] Benjamin Nagel, Christian Gerth, Gregor Engels, and Jennifer Post. Ensuring consistency among business goals and business process models. In *2013 17th IEEE International Enterprise Distributed Object Computing Conference*, pages 17–26. IEEE, 2013.

[12] Gerd Gröner, Mohsen Asadi, Bardia Mohabbati, Dragan Gašević, Fernando Silva Parreiras, and Marko Bošković. Validation of user intentions in process models. In *International Conference on Advanced Information Systems Engineering*, pages 366–381. Springer, 2012.

[13] Axel Van Lamsweerde. *Requirements engineering: From system goals to UML models to software*, volume 10. Chichester, UK: John Wiley & Sons, 2009.

[14] Jennifer Horkoff, Fatma Başak Aydemir, Evellin Cardoso, Tong Li, Alejandro Maté, Elda Paja, Mattia Salnitri, Luca Piras, John Mylopoulos, and Paolo Giorgini. Goal-oriented requirements engineering: an extended systematic mapping study. *Requirements Engineering*, 24(2):133–160, 2019.

[15] Hiroki Horita, Hideaki Hirayama, Yasuyuki Tahara, and Akihiko Ohsuga. Towards goal-oriented conformance checking. In *The 27th International Conference on Software Engineering and Knowledge Engineering, SEKE 2015*, pages 722–724, 2015.

[16] Hiroki Horita, Hideaki Hirayama, Takeo Hayase, Yasuyuki Tahara, and Akihiko Ohsuga. A method for goal model repair based on process mining. In *2019 20th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pages 121–126. IEEE, 2019.

[17] Jiaqi Yan, Daning Hu, Stephen S Liao, and Huaiqing Wang. Mining agents' goals in agent-oriented business processes. *ACM Transactions on Management Information Systems (TMIS)*, 5(4):1–22, 2014.

[18] Renata Guizzardi and Ariane Nunes Reis. A method to align goals and business processes. In *International Conference on Conceptual Modeling*, pages 79–93. Springer, 2015.

[19] Andrey V Skobtsov and Anna A Kalenkova. Using heuristic algorithms for fast alignment between business processes and goals. In *2019 IEEE 23rd International Enterprise Distributed Object Computing Workshop (EDOCW)*, pages 85–91. IEEE, 2019.

[20] Mari Abe and Michiharu Kudo. Analyzing business processes by automatically detecting kpi thresholds. In *2016 IEEE International Conference on Services Computing (SCC)*, pages 187–194. IEEE, 2016.

[21] Jacek Dabrowski, Fitsum Meshesha Kifetew, Denisse Muñante, Emmanuel Letier, Alberto Siena, and Angelo Susi. Discovering requirements through goal-driven process mining. In *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, pages 199–203. IEEE, 2017.

[22] Karthikeyan Ponnalagu, Aditya Ghose, Nanjangud C Narendra, and Hoa Khanh Dam. Goal-aligned categorization of instance variants in knowledge-intensive processes. In *International Conference on Business Process Management*, pages 350–364. Springer, 2016.

[23] Marcus Dees, Massimiliano de Leoni, and Felix Mannhardt. Enhancing process models to improve business performance: A methodology and case studies. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 232–251. Springer, 2017.

[24] Artem Polyvyanyy, Wil MP Van Der Aalst, Arthur HM Ter Hofstede, and Moe T Wynn. Impact-driven process model repair. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 25(4):1–60, 2016.

[25] Abel Armas Cervantes, Nick RTP van Beest, Marcello La Rosa, Marlon Dumas, and Luciano García-Bañuelos. Interactive and incremental business process model repair. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 53–74. Springer, 2017.

[26] Andrea Burattin. Plg2: Multiperspective process randomization with online and offline simulations. In *BPM (Demos)*, pages 1–6. Citeseer, 2016.