Identifying the Key Attributes in an Unlabeled Event Log for Automated Process Discovery

Kentaroh Toyoda, *Member, IEEE*, Rachel Gan Kai Ying, Allan NengSheng Zhang, and Tan Puay Siew

✦

Abstract—Process mining discovers and analyzes a process model from historical event logs. The prior art methods use the key attributes of case-id, activity, and timestamp hidden in an event log as clues to discover a process model. However, a user needs to specify them manually, and this can be an exhaustive task. In this paper, we propose a two-stage key attribute identification method to avoid such a manual investigation, and thus this is a step toward fully automated process discovery. One of the challenging tasks is how to avoid exhaustive computation due to combinatorial explosion. For this, we narrow down candidates for each key attribute by using supervised machine learning in the first stage and identify the best combination of the key attributes by discovering process models and evaluating them in the second stage. Our computational complexity can be reduced from $\mathcal{O}(N^3)$ to $\mathcal{O}(k^3)$ where N and k are the numbers of columns and candidates we keep in the first stage, respectively, and usually k is much smaller than N. We evaluated our method with 14 open datasets and showed that our method could identify the key attributes even with $k = 2$ for about 20 seconds for many datasets.

Index Terms—Process mining, key attribute identification, automated process discovery

1 INTRODUCTION

Process mining enables us to discover and analyze a business process model from historical event logs [1]. In particular, process discovery, as its name suggests, is the step to discover a business process model by finding a pattern of sequential processes in an event log. Once a process model is obtained, we can use it for further analysis such as conformance check (i.e., identifying whether a newly given event log follows the discovered process model) and bottleneck analysis (i.e., identifying which parts of processes are bottlenecks and can be improved). They have been proven to be successful in many industries, including supply chain (e.g., [2]) and manufacturing (e.g., [3], [4], [5]).

Process discovery algorithms, often called miners, have been well-studied in the last two decades (e.g., [6], [7], [8],

Kentaroh Toyoda is with the Institute of High Performance Computing (IHPC), Agency for Science, Technology and Research (A*STAR), 1 Fusionopolis Way, #16-16 Connexis, Singapore 138632, Republic of Singapore. E-mail: kentaroh.toyoda@ieee.org
Rachel Gan Kai Ying is with Nanyang Technological University (NTU), Republic of Singapore.
Allan NengSheng Zhang, and Tan Puay Siew are with Singapore Institute of Manufacturing Technology (SIMTech), Agency for Science, Technology and Research (A*STAR), 5 CleanTech Loop #01-01, CleanTech Two Block B, Singapore 636732, Republic of Singapore.
The research was conducted when K. Toyoda and Gan K. Y. R. were in SIMTech.

[9], [10], [11]). The basic idea of process discovery is to find the frequent patterns of ordered and parallel execution of processes. Most of the existing algorithms discover a process model using three attributes in an event log, namely (i) case-id, (ii) timestamp, and (iii) activity (hereafter, the key attributes). Processes with the same case-id are a series of correlated processes. A timestamp is used in process discovery to know the order of processes in the same case-id, and activity is the name of a process. Fig. 1 shows an example of an event log in a hamburger shop and its discovered process model.[1] In this example, the ID can be seen as the case-id as the processes in the same case-id seem to be correlated. Also, the activity and datetime columns are used as the activity and timestamp, respectively.

Most of the existing process discovery algorithms require users to specify the key attributes in an event log. However, such a manual approach is not desirable when an event log contains dozens of or even hundreds of columns. Furthermore, as process mining may not necessarily be executed by practitioners, an automated process discovery where a user does not have to specify the key attributes in an event log would be appreciated in many situations.

In this line of work, Abbad Andaloussi et al. proposed a method to identify a case-id column in an event log [12]. Their idea is to test every column in the event log as the case-id attribute and evaluate a process model discovered by such an assumption. However, this method only identifies the case-id attribute. Second, its computation is exhaustive as it repeatedly discovers and evaluates process models for every case-id candidate. If we were to identify case-id, timestamp, and activity attributes with their method, the computation cost would be $\mathcal{O}(N^3)$ where N is the number of columns of a given event log.

This paper considers the scenario where raw data are given in a table format and the columns of case-ids, activity, and timestamp are hidden in the table. The objective is to identify these attributes efficiently. More specifically, let L be a table-formatted raw log data with N_{rows} rows and N_{attr} attributes, i.e., $A = \{a_1, a_2, \cdots, a_{N_{\text{attr}}}\}$ where $N_{\text{attr}} \geq 3$. Furthermore, L consists of raw event values $E = \{\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_{N_{\text{rows}}}\}$ where $\mathbf{e}_i = \{v_1^i, v_2^i, \cdots, v_{N_{\text{attr}}}^i\}$. The objective is to identify (i) case-id, (ii) activity, and (iii) timestamp attributes in A by analyzing E. In this setting,

1. We modified the example event log in https://pm4py.fit.fraunhofer.de/getting-started-page
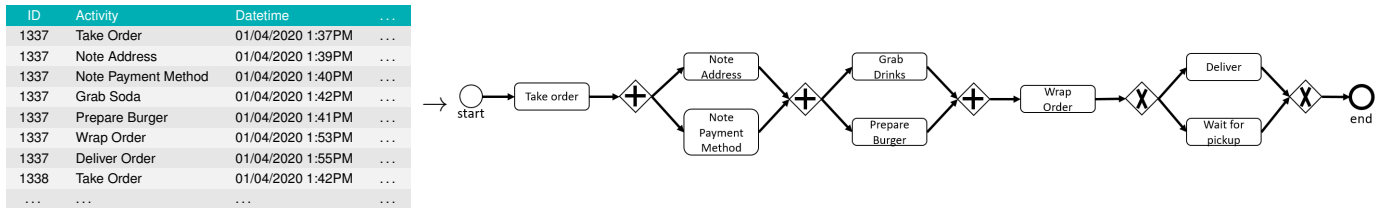
Fig. 1: **Left:** An example of event logs; **Right:** Process model discovered from the table.

we propose a two-stage key attribute identification method for automated process discovery. To avoid exhaustive computation due to combinatorial explosion, we first narrow down candidates for each key attribute by using supervised machine learning. If we could identify a single candidate for each key attribute, then we output them as the key attributes. Otherwise, we identify the best combination of the candidates by discovering and evaluating a process model for every combination. We define five scoring functions for this evaluation. Our computational complexity can be reduced from $\mathcal{O}(N^3)$ to $\mathcal{O}(k^3)$ where k is the number of candidates we keep in the first stage, where usually k is much smaller than N. We thoroughly evaluated our method with 14 open datasets, clarified which parameters affect accuracy and computation, and showed that our method could feasibly identify the key attributes even with $k = 2$ within a reasonable time (about 20 seconds).

The contributions of our research are as follows:

1) To the best of our knowledge, we are the first to identify all the key attributes in an event log.
2) We propose a two-stage method to avoid the combinatorial explosion to reduce our computational complexity from $\mathcal{O}(N^3)$ to $\mathcal{O}(k^3)$ where $k \ll N$.

The remainder of this paper is organized as follows. Section 2 summarizes the prior art and the motivation of our work. Section 3 describes the proposed method. Section 4 shows the results of the performance evaluation. Section 5 discusses possible future work. Section 6 concludes the paper.

2 RELATED WORK

2.1 Process Discovery

The mainstream research is the design of process discovery algorithms or miners. Process discovery is a method to extract the ordered sequences of processes from an event log. As it is infeasible to cover all the process discovery algorithms due to space limitations, we introduce some well-studied ones here. According to a recent survey paper [13], Cook and Wolf proposed the first process discovery algorithm based on a finite-state machine (FSM) in 1995 [6]. van der Aalst proposed to apply Petri-nets to process mining, which is often called Alpha miner, in 2002 [7]. Weijters and van der Aalst extended Alpha miner and proposed a new algorithm called heuristics miner (HM) to handle noisy event logs by taking into account the frequency of dependent activities [8]. It also handles possible short loops that cannot be dealt with in Alpha miner. HM is often chosen as it has been proven to be successful in real-event logs (e.g., [14]). Hence, it is often used as a base miner as in [10]. vanden Broucke

and De Weerdt solved several issues in HM and proposed an extended algorithm called Fodina [10]. Leemans et al. proposed a series of inductive miners (IMs) [9], [15], [16]. IM discovers a set of block-structured process models to ensure that the discovered models are sound and fit the observed behaviour [9]. Leemans et al. extended the original IM to filter out infrequent behavior quickly [15]. They further extended IM to make it scalable by discovering a directly-follows graph (DFG) from an event log once and applying a divide-and-conquer strategy [16]. Augusto et al. proposed a split miner (SM) [17]. SM discovers a process model by filtering a DFG induced by an event log and identifying the combinations of split gateways that capture the concurrency, conflict, and causal relations between neighbors in the DFG.

2.2 Pre-processing Raw Data

The above process discovery methods can be applied to event logs with explicit case-ids, timestamps, and activities. However, raw data, which are often extracted from systems such as customer relationship management (CRM) and enterprise resource planning (ERP), are not yet ready for event logs. Diba et al. argue that we typically need to follow three steps to generate event logs from raw data, which are (i) event extraction, (ii) event correlation, and (iii) event abstraction [18].

The event extraction is a process to retrieve raw event logs from databases. The event correlation is a step to correlate events by cases in the extracted logs. There could be cases where we need to correlate events (i) when raw data is missing case-ids, which correlate the series of events in a log, and (ii) when case-ids are hidden in raw data and unknown. The event correlation techniques have been well studied to solve the former issue. An efficient approach is key, as there could be many possible combinations for correlating events in such unlabelled raw data. For instance, Ferreira and Gillblad proposed a Markov-chain-based approach to infer a process model leveraging the activity transition patterns [19]. Motahari-Nezhad et al. proposed efficient algorithms and heuristics to discover correlated events that could interest users [20]. Bayomie et al. proposed a series of case-id inference algorithms with a complete process model (e.g., [21][22]). Their approaches find the best event correlation via optimization. Specifically, they try to minimize the misalignment between a generated event log and an input process model and the activity execution time variance across cases using a simulated annealing algorithm. Helal and Awad proposed an online approach to handle streaming raw data [23]. Reguieg et al. proposed a parallelized algorithm with MapReduce to scale the event correlation process [24].

For the cases when case-ids are hidden in raw data and unknown, few approaches have been investigated. Abbad Andaloussi et al. proposed an unsupervised case-id identification method [12]. It identifies a case-id column in a given event log when it is not specified. Its idea is to find an attribute that seems most likely to be the case-id by applying a process discovery algorithm to every case-id candidate column and measuring its likeliness based on performance metrics such as recall and simplicity. Its rationality comes from the fact that a process model discovered with the correct case-id should give us better performance than that discovered with a wrong case-id column. Bala et al. proposed a method to identify the case-id and activity columns in a raw event table [25]. They leveraged the fact that we should see some repetitiveness of case-ids and activity columns. However, repetitiveness alone detects case-ids falsely. Hence, they considered the pairs of events with high individual repetitiveness but a low pairwise repetitiveness to reduce such a false detection.

2.3 Motivation of Our Work

Most of the existing process discovery algorithms require users to specify the key attributes in an event log, which would be problematic in some situations. First, it is tedious to manually look for these attributes as an event log is expected to contain tens of attributes. Furthermore, as process mining would be conducted not only by practitioners but also by laypeople, an automated approach is preferable, where a user does not have to specify attributes to be used as the key attributes. Our goal is fully automated process mining, where no human intervention is required to discover a meaningful process flow from an event log.

Abbad Andaloussi et al.'s work [12] would be the closest to our goal. However, there are challenges to truly automated process mining. First, it only identifies a case-id attribute, not a timestamp or activity. Second, its computation is exhaustive as it repeats process discovery to every case-id candidate. If we are to identify case-id, timestamp, and activity attributes at once, the second challenge would be a big issue. Hence, our work aims to identify all the key attributes in an event log without incurring exhaustive computation.

3 PROPOSED METHOD

We propose a two-stage key attribute identification method. As an event log may have many attributes, we first narrow down candidates by analyzing the characteristics of values in each key attribute. If we could identify a single candidate for each key attribute, we output them as the key attributes. Otherwise, we identify the best candidate by evaluating every combination of key attribute candidates using a process discovery algorithm. We choose the candidate tuple with the highest score as the key attributes.

Fig. 2 illustrates our method. Our assumption on the input event log is that it surely contains the key attributes, but they are not explicitly given.

3.1 Narrow down candidates

We first narrow down the candidates of the key attributes against a given unlabeled event log. Figure 3 shows the overview of the procedures of this step.

3.1.1 Feature extraction

We leverage the fact that there are noticeable characteristics in each key attribute. For instance, for case-ids, the same case-ids should appear multiple times in an attribute, otherwise, we cannot extract a process model with a single process. Similarly, for timestamps, we can expect many digits and symbols such as '-' and ':' in the values. To capture such characteristics, we extract the nine features listed in TABLE 1 from each column.[2] We call the first top seven features in the list local features and the last two global features. The local features are to extract characteristic features from each attribute value. For instance, for timestamps, we would expect more digits and symbols such as '-' and ':' than alphabets in the values. In contrast, the global features are to extract the frequency of the same values in a column. For instance, we expect the same values to appear multiple times for a case-id column. As each local feature is a vector calculated from a value in an attribute, we calculate their mean values as the final local features. For instance, if the local features of an attribute contain three values, {Case-01, Case-02, Case-03}, as each value involves three small letters, i.e. 'a', 's', and 'e', out of six characters, $f_{\text{s.letters}} = (3/6 + 3/6 + 3/6)/3 = 0.5$.

In contrast, global features are features extracted from values in an attribute. For instance, if values in an attribute are $\{1, 2, 3, 1, 2\}$, three unique values, namely 1, 2, and 3, appear twice, twice, and once, $f_{\text{r.unique}} = 3/5 = 0.6$ and $f_{\text{m.unique}} = (2 + 2 + 1)/3 = 1.67$, respectively.

3.1.2 Classification

We then leverage supervised machine learning with the above features to narrow down the candidates for the key attributes. We chose supervised machine learning because many labeled event logs are available online.[3] We build three binary classifiers that output an attribute's probabilities of case-id, timestamp, and activity. For an unlabeled event log, we calculate the above features of each column and input them to trained classifiers to obtain the probabilities of whether an attribute belongs to case-id, timestamp, or activity. We choose the column(s) with the N_{top} highest probability as candidates for each key attribute. The larger N_{top} is, the more candidates we would obtain. For instance, when $N_{\text{top}} = 3$, we will at least have $3^3 = 27$ candidates. Hence, the possible N_{top} would be one or two. We will evaluate the relationships between N_{top}, accuracy, and computation time in Section 4. If $N_{\text{top}} = 1$ and we have only one single column for each key attribute, we output them as the key attributes and do not proceed to the next stage. Otherwise, i.e., when more than one column has the highest probability, we need to identify the best one through the second stage, which we will explain in the next section.

3.2 Discover and evaluate process models

When having more than one candidate, we discover and evaluate a process model for each and determine the best one based on a performance metric. For an event log labeled by each candidate, we divide it into two parts by a case-id column (candidate) and evaluate the goodness of the chosen

---

2. Note that we only use the first N_{rows}, say 1,000, rows for feature extraction to reduce computation time.

3. E.g., http://www.processmining.org/event-data.html

1. Input an unlabeled event log

A given event log may not have explicit column names.

0	1	...	N
E_1	2022-08-01 13:09:40	...	John
E_1	2022-08-01 13:30:23	...	-
E_1	2022-08-01 15:03:12	...	Alice
E_2	2022-08-02 09:01:29	...	John
...		...	...

2. Narrow down candidates

Heuristics and machine learning will narrow down the candidates of case_ids, timestamps, and activities.

	case_id cand	timestamp cand	activity cand
1	0	1	4
2	0	1	5
3	0	2	4
4	0	2	5
...	...	...	...

3. Discover and evaluate process models

A process model is discovered and evaluated for every candidate. The candidate with the highest score will be chosen as the correct key columns.

	c	t	a	process model	score
1	0	1	4		0.85
2	0	1	5		**0.96**
3	0	2	4		0.67
4	0	2	5		0.74
...	...	...	...	...	...

Fig. 2: Overview of our method.

TABLE 1: Extracted features.

FEATURE	DEFINITION
$f_{s.letters}$	Ratio of small letters
$f_{l.letters}$	Ratio of large letters
f_{digits}	Ratio of digits
f_{spaces}	Ratio of spaces (including tabs)
$f_{symbols}$	Ratio of symbols
f_{chars}	Number of characters
f_{words}	Number of words in a value
$f_{r.unique}$	Ratio of unique values
$f_{m.unique}$	Mean of number of each unique value

1. Feature extraction

0	1	...	N
E_1	2022-08-01 13:09:40	...	John
E_1	2022-08-01 13:30:23	...	-
E_1	2022-08-01 15:03:12	...	Alice
E_2	2022-08-02 09:01:29	...	John
...		...	...

Feature	0	1	...	N
$f_{s.letters}$	0	0	...	0.81
$f_{l.letters}$	0.33	0	...	0.11
...	...	...	...	...
$f_{r.unique}$	0.31	0.19	...	0.32
$f_{m.unique}$	4.2	1.1	...	3.4

Classifier (case-id)

Classifier (timestamp)

Classifier (activity)

2. Classification

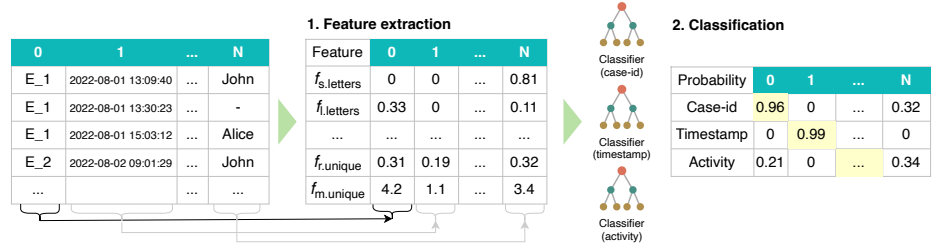| Probability	0	1	...	N
Case-id	0.96	0	...	0.32
Timestamp	0	0.99	...	0
Activity	0.21	0	...	0.34

Fig. 3: Overview of our first stage.

attributes with 2-fold cross-validation (CV). Specifically, we first use the first part for process model discovery and the second part for evaluation, and then swap the parts and do the same evaluation again. For instance, when a case-id candidate consists of $\{1, 2, \cdots, 9, 10\}$, we first discover the process model with the event log that contains $\{1, 2, \cdots, 5\}$ and evaluate it with the event log containing $\{6, 7, \cdots, 10\}$. We then swap and evaluate the training and evaluation event logs and calculate the average of these two scores as the final score.

We can use any process discovery algorithms for this (e.g., [8], [9], [15]) and performance metrics (e.g., fitness, precision, generalization, simplicity, and their combinations [26], [12]). Fitness indicates how much of the observed behavior in an event log fits (or explains) the discovered process model. Precision, in contrast, quantifies how fewer unnecessary possible paths are generated by the discovered model. Generalization is to quantify how the discovered model has the flexibility to an unseen event log. Last but not least, structure or simplicity quantifies how fewer nodes are used in the discovered model.

In some cases, a process discovery algorithm takes a long time to discover and evaluate a process model or even fails to do so. This would not happen in the case where the combination of the key attributes is correct but would happen when it cannot find similarities among cases due to the wrong choice of the combination. To further save computation time, we introduce two thresholds, th_{dis} and th_{eval}, to simply stop discovering or evaluating a process model and return 0 as a score after th_{dis} or th_{eval} has passed for process discovery or evaluation. These parameters have to be tuned based on the computer's specifications.

4 PERFORMANCE EVALUATION

We conducted a performance evaluation to clarify the validity of our method. We evaluated accuracy and computation time against the recent business process intelligence challenge (BPIC) datasets listed in TABLE 2. We used the BPIC datasets that explicitly contain the key attributes (i.e., "case:concept:name" for case-id, "time:timestamp" for timestamp, and "concept:name" for activity).[4][5] Accuracy was measured for each key attribute and defined as the number of correctly identified cases divided by the number of trials. Computation time was measured by the stages. As the first stage includes supervised machine learning, we executed a hold-out validation where one dataset is kept for testing while the others are used for training. We evaluated the accuracy and computation time of each dataset 10 times each. Our method is miner-agnostic, meaning any mining algorithm can be used, and thus we referred to Augusto et al.'s seminal benchmark paper [11] to choose the miners for evaluation. According to [11], although there is no dominant miner, inductive miner (IM) is found to be one of the most accurate miner. In contrast, heuristics miner (HM) is not the best or the latest algorithm, but we would like to see how the difference in miners affects the final result of our method. Hence, we tested the two process discovery algorithms, IM [9] and HM [8], in the second

4. BPIC 2014 and 2016 datasets do not contain these attributes, and we thus did not use them.

5. Although we used the explicitly labeled key attributes as unique correct tuples for the performance evaluation purpose, other tuples could be sufficient for users. From this point of view, our first step may not be sufficient for the evaluation purpose where there is an assumption that only a single correct set of tuples exists. However, in reality, there could be more than one set of key attributes on the table that could be used as an activity, a case-id, or a timestamp [20], and we are technically not able to identify one tuple.

TABLE 2: Datsets used in our evaluation.

Dataset	#Columns
BPIC2011	128
BPIC2012	7
BPIC2013 incident management	12
BPIC2013 problem management (closed)	12
BPIC2013 problem management (open)	11
BPIC2015 1	29
BPIC2015 2	28
BPIC2015 3	29
BPIC2015 4	29
BPIC2015 5	29
BPIC2017	19
BPIC2018	75
BPIC2019	21
BPIC2020	22

TABLE 3: Parameters of the process discovery algorithms.

Miner	Parameter	Value
IM	Noise threshold	0.2
HM	Dependency threshold	0.5
HM	AND measure threshold	0.65
HM	Minimum number of occurrences of activity	1
HM	Minimum DFG occurrences	1
HM	DFG pre-cleaning noise threshold	0.05
HM	Loop length two threshold	2

TABLE 4: Breakdown comparison of computation time.

k	Miner	Computation time [s]	
		Stage 1	Stage 2
1	HM	0.58	5.65
1	IM	0.58	17.15
2	HM	0.60	17.01
2	IM	0.60	43.68

stage, and their parameters are listed in TABLE 3, which are all default parameter settings defined by their libraries. We implemented and evaluated our method using Python and its process mining library, PM4Py [27].[6] We extracted the first $N_{\text{rows}} = 1,000$ rows for the evaluation for each dataset. We also set $th_{\text{dis}} = 5$ seconds and $th_{\text{eval}} = 60$ seconds. The reason behind the choice of N_{rows} is that we wanted to streamline the condition of the evaluation and to make sure the computation finishes in a reasonable time. Likewise, th_{dis} and th_{eval} were heuristically chosen so that we did not miss the correct combination of key attributes.

Let us first discuss how miners used in the second stage affect the accuracy and computation time. Fig. 4 and TABLE 4 show the accuracy and computation time to compare miners. As can be seen from Fig. 4, we do not have a miner that dominates the other. When we average the three types of accuracy, IM is slightly better than HM; however, it is not significant. However, when we compare computation time by the miners, as in TABLE 4, the difference is significant. HM is 2-3 times faster than IM regardless of k. As can be seen from the table, the computationally dominant part resides in the second stage to discover a process model and evaluate it. We are interested in how the metric used in the evaluation part of the second stage affects the computation time and accuracy. Fig. 5 shows the computation time by the metrics. Buijs et al.'s weighted average of the four individual metrics [26], which we call Buijs2014, is the most time-consuming because it computes all four metrics. Among the four fundamental metrics, we found that precision is the most time-consuming metric. When we look at the accuracy results in Fig. 6, we cannot see a significant difference in terms of accuracy, but generalization achieves the best accuracy among the five metrics. From these results, we can say that generalization is the most balanced metric in the second stage. We then discuss how to choose k. Technically, the first stage should leave the correct key attributes. We evaluated a metric of coverage, whether the first step can successfully retain the correct key attributes. We measured the coverage metric by counting the cases where the correct key attributes were kept in the first step and dividing it by the total cases for $k = 1$ and 2. We evaluated it against all the datasets and averaged the results. Fig. 8 shows the results of coverage versus k. We can see from this figure that the larger k the more correct key attributes we can retain. However, we filtered out the correct key attributes, in particular, case IDs and activities, even with $k = 4$.

6. Our source code including dataset preparation, evaluation, and result summarization is available at https://github.com/kentaroh-toyoda/research-process-mining.

The larger k, the more coverage we can achieve; however, from TABLE 4, the larger k the more computation time. This is because we need to evaluate more combinations of candidates for a large k. Hence, we want to set k as small as possible. Luckily, we do not see much improvement when we increase k from 1 to 2 as can be seen from Fig. 7.

Note that even if we set $k = 1$, meaning that when we try to leave only the single best candidate for each key attribute in the first stage, the second stage still does a job as can be seen from TABLE 4. This means that we faced situations where there were multiple best candidates in the first stage.

Let us discuss the relationships between accuracy and the number of attributes in a dataset. Fig. 9 shows the averaged accuracy versus the number of attributes in the datasets. There is a negative correlation between accuracy and the number of attributes. We can explain this with a random guess where one randomly guesses which column should be used as case-id, activity, and timestamp. If we have 20 attributes, the accuracy of random guess would be roughly $0.05(= 1/20)$ for each attribute. However, this decreases to 0.01 when a dataset contains 100 attributes.

5 OPEN PROBLEMS

As the evaluation above shows, our method needs more polishment in terms of accuracy and computation time, in particular, when a given dataset is large. How well we can narrow down candidates in the first stage would be key to improving accuracy and reducing computation time at the same time. Our current approach is to judge the possibility of each key attribute against each table column. However, one possible solution is to take into account information captured from multiple columns or even from a whole table. Although their work's objective is not exactly the same as ours, Zhang et al. proposed to consider the relationships between adjacent columns to identify each table column's type [28]. Another promising direction would be to incorporate a context-aware approach (e.g., [29], [30], [31]). If we could capture the context of the event log, then we would be able to find more suitable key attribute candidates.

Regarding the computation time, one of the possible approaches is partially evaluating models with sampled event logs instead of using the whole one in the second stage. We will need to make sure how many samples are enough to assess the models. Another possibility to reduce computation time would be to polish our implementation. For instance, making use of a graphics processing unit (GPU) would help speed up the computation [32].
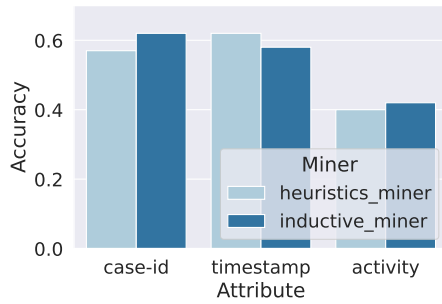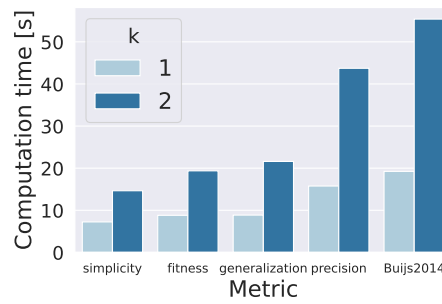
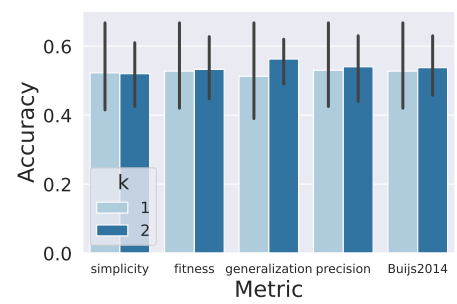Fig. 4: Accuracy versus miners.   Fig. 5: Computation time versus metrics.   Fig. 6: Accuracy versus metrics.
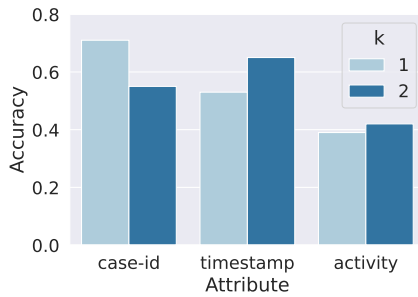
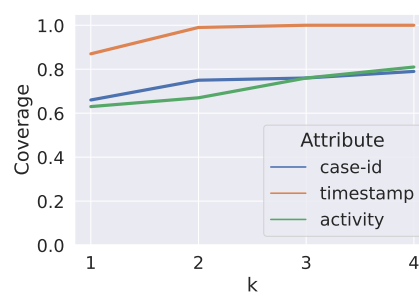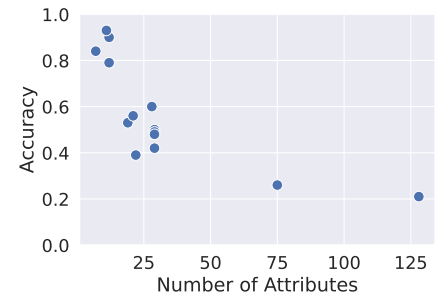Fig. 7: Accuracy versus attributes.   Fig. 8: Coverage versus k.   Fig. 9: Accuracy versus the number of attributes in datasets.

6 CONCLUSIONS

We have proposed a two-stage key attribute identification method for automated process discovery. If we were to evaluate all possible key attributes, their combinations would be too huge to try. To avoid such exhaustive computation, we first narrow down candidates by analyzing the characteristics of values in each column. If we could identify a single candidate for each key attribute, we output them as the key attributes. Otherwise, we identify the best candidate by evaluating every combination of key attribute candidates using a process discovery algorithm in the second stage. We chose the candidate tuple with the highest probabilities as the key attributes. We evaluated our method with 14 open datasets and showed that our method could feasibly identify key attributes for the datasets with fewer columns within a reasonable time.

To the best of our knowledge, this paper is the first to identify the key attributes in an event log. Hence, we believe that the proposed method still has room for improvement. We have discussed the open problems and suggested several possible research directions toward fully reliable automated process discovery.

ACKNOWLEDGMENT

REFERENCES

[1] W. Van Der Aalst, "Process mining," *Communications of the ACM*, vol. 55, no. 8, pp. 76–83, Aug. 2012.

[2] Jokonowo, Claes, Sarno, and others, "Process mining in supply chains: a systematic literature review," *International journal of high risk behaviors & addiction*, 2018.

[3] E. R. Mahendrawathi, N. Arsad, H. M. Astuti, R. P. Kusumawardani, and R. A. Utami, "Analysis of production planning in a global manufacturing company with process mining," *Journal of Enterprise Information Management*, vol. 31, no. 2, pp. 317–337, Mar. 2018.

[4] E. Ruschel, E. A. P. Santos, and E. d. F. R. Loures, "Establishment of maintenance inspection intervals: an application of process mining techniques in manufacturing," *Journal of intelligent manufacturing*, vol. 31, no. 1, pp. 53–72, Jan. 2020.

[5] R. Lorenz, J. Senoner, W. Sihn, and T. Netland, "Using process mining to improve productivity in make-to-stock manufacturing," *International Journal of Production Research*, vol. 59, no. 16, pp. 4869–4880, 2021.

[6] J. E. Cook and A. L. Wolf, "Automating Process Discovery through Event-Data Analysis," in *Proc. of IEEE International Conference on Software Engineering*, Apr. 1995, pp. 73–73.

[7] W. M. P. van der Aalst, "Making Work Flow: On the Application of Petri Nets to Business Process Management," in *Proc. of Application and Theory of Petri Nets 2002*. Springer Berlin Heidelberg, 2002, pp. 1–22.

[8] Weijters, Der Aalst, and others, "Process mining with the heuristics miner algorithm," *Eindhoven, Tech. Rep*, 2006.

[9] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Discovering Block-Structured Process Models from Event Logs - A Constructive Approach," in *Proc. of Application and Theory of Petri Nets and Concurrency*. Springer Berlin Heidelberg, 2013, pp. 311–329.

[10] S. K. L. M. vanden Broucke and J. De Weerdt, "Fodina: A robust and flexible heuristic process discovery technique," *Decision support systems*, vol. 100, pp. 109–118, Aug. 2017.

[11] A. Augusto, R. Conforti, M. Dumas, M. La Rosa, and A. Polyvyanyy, "Split miner: automated discovery of accurate and simple business process models from event logs," *Knowledge and information systems*, vol. 59, no. 2, pp. 251–284, May 2019.

[12] A. Abbad Andaloussi, A. Burattin, and B. Weber, "Toward an Automated Labeling of Event Log Attributes," in *Proc. of Enterprise, Business-Process and Information Systems Modeling*. Springer International Publishing, 2018, pp. 82–96.

[13] C. d. S. Garcia, A. Meincheim, E. R. Faria Junior, M. R. Dallagassa,

D. M. V. Sato, D. R. Carvalho, E. A. P. Santos, and E. E. Scalabrin, "Process mining techniques and applications – A systematic mapping study," *Expert systems with applications*, vol. 133, pp. 260–295, Nov. 2019.

[14] J. De Weerdt, M. De Backer, J. Vanthienen, and B. Baesens, "A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs," *Information systems*, vol. 37, no. 7, pp. 654–676, Nov. 2012.

[15] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Discovering Block-Structured Process Models from Event Logs Containing Infrequent Behaviour," in *Proc. of Business Process Management Workshops*. Springer International Publishing, 2014, pp. 66–78.

[16] S. J. J. Leemans, D. Fahland, and W. M. P. Van Der Aalst, "Scalable process discovery with guarantees," *Lecture Notes in Business Information Processing*, vol. 214, pp. 85–101, 2015.

[17] A. Augusto, M. Dumas, M. La Rosa, S. J. J. Leemans, and S. K. L. M. vanden Broucke, "Optimization framework for DFG-based automated process discovery approaches," *Software and Systems Modeling*, pp. 1–26, 2021.

[18] K. Diba, K. Batoulis, M. Weidlich, and M. Weske, "Extraction, correlation, and abstraction of event data for process mining," *Wiley interdisciplinary reviews. Data mining and knowledge discovery*, vol. 10, no. 3, p. e1346, 2020.

[19] D. R. Ferreira and D. Gillblad, "Discovering process models from unlabelled event logs," in *Proc. of Business Process Management*. Springer Berlin Heidelberg, 2009, pp. 143–158.

[20] H. R. Motahari-Nezhad, R. Saint-Paul, F. Casati, and B. Benatallah, "Event correlation for process discovery from web service interaction logs," *The VLDB journal*, vol. 20, no. 3, pp. 417–444, Jun. 2011.

[21] D. Bayomie, C. Di Ciccio, M. La Rosa, and J. Mendling, "A probabilistic approach to Event-Case correlation for process mining," in *Proc. of Conceptual Modeling*. Springer International Publishing, 2019, pp. 136–152.

[22] D. Bayomie, K. Revoredo, C. Di Ciccio, and J. Mendling, "Improving accuracy and explainability in Event-Case correlation via rule mining," in *Proc. of International Conference on Process Mining (ICPM)*. IEEE, Oct. 2022, pp. 24–31.

[23] I. M. A. Helal and A. Awad, "Online correlation for unlabeled process events: A flexible CEP-based approach," *Information systems*, vol. 108, p. 102031, Sep. 2022.

[24] H. Reguieg, B. Benatallah, H. R. M. Nezhad, and F. Toumani, "Event correlation analytics: Scaling process mining using Mapreduce-Aware event correlation discovery techniques," *IEEE Transactions on Services Computing*, vol. 8, no. 6, pp. 847–860, Nov. 2015.

[25] S. Bala, J. Mendling, M. Schimak, and P. Queteschiner, "Case and activity identification for mining process models from middleware," in *Proc. of The Practice of Enterprise Modeling*. Springer International Publishing, 2018, pp. 86–102.

[26] J. C. A. M. Buijs, B. F. van Dongen, and W. M. P. van der Aalst, "Quality Dimensions in Process Discovery: The Importance of Fitness, Precision, Generalization and Simplicity," *International Journal of Cooperative Information Systems*, vol. 23, no. 01, p. 1440001, Mar. 2014.

[27] A. Berti, S. J. van Zelst, and W. van der Aalst, "Process mining for python (PM4Py): bridging the gap between process-and data science," *arXiv preprint arXiv:1905.06169*, 2019.

[28] D. Zhang, Y. Suhara, J. Li, M. Hulsebos, Ç. Demiralp, and W.-C. Tan, "Sato: Contextual semantic type detection in tables," *Proc. of VLDB Endowment*, vol. 13, no. 12, p. 1835–1848, 2020. [Online]. Available: https://doi.org/10.14778/3407790.3407793

[29] Z. Mounira and B. Mahmoud, "Context-aware process mining framework for Business Process flexibility," in *Proc. of International Conference on Information Integration and Web-based Applications & Services (iiWAS)*. ACM, 2010, pp. 421–426.

[30] R. Song, J. Vanthienen, W. Cui, Y. Wang, and L. Huang, "Towards a comprehensive understanding of the context concepts in context-aware business processes," in *Proc. of International Conference on Subject-Oriented Business Process Management*, 2019, pp. 1–10.

[31] J. vom Brocke, M.-S. Baier, T. Schmiedel, K. Stelzl, M. Röglinger, and C. Wehking, "Context-Aware Business Process Management," *Business & Information Systems Engineering*, pp. 1–18, 2021.

[32] A. Berti, M. P. Nghia, and W. M. P. van der Aalst, "PM4Py-GPU: A High-Performance General-Purpose Library for Process Mining," in *Proc. of Research Challenges in Information Science*. Springer International Publishing, 2022, pp. 727–734.

Kentaroh Toyoda was born in Tokyo, Japan, in 1988. He received B.E., M.E., and Ph.D. (Engineering) degrees from the Department of Information and Computer Science, Keio University, Japan, in 2011, 2013, and 2016, respectively. He was a scientist at the Singapore Institute of Manufacturing Technology (SIMTech), A*STAR, from Apr. 2019 to Oct. 2022 and is currently a senior scientist at the Institute of High Performance Computing (IHPC), A*STAR, Singapore. His research interests include blockchain, security and privacy, data analysis, and mechanism design.

Gan Kai Ying Rachel received a B.E. (Engineering) in Information Systems Technology and Design from the Singapore University of Technology and Design, Singapore in 2019. She was a research engineer at the Singapore Institute of Manufacturing Technology (SIMTech), A*STAR, from 2020 to 2022. She is pursuing her master's degree in AI at Nanyang Technological University (NTU), Singapore. Her research interests include data analytics, machine learning, and their applications.

Allan NengSheng Zhang is a Senior Pricipal Scientist with SIMTech, A*STAR, Singapore. He has more than 25 years of experience in the development of operations technologies (OT) using AI such as knowledge-based systems and enterprise information systems development. He is presently an Adjunct Associate Professor with the School of School of Mechanical and Aerospace Engineering, at Nanyang Technological University. His research interests include knowledge management, data mining, machine learning, artificial intelligence, computer security, software engineering, software development methodology and standards, and enterprise information systems. He and his group are currently working toward research in cyber-physical system modeling, manufacturing system analyses including data mining, supply chain information management, supply chain risk and resilience management using a complex systems approach, multi-objective vehicle routing problems, and urban last-mile logistics.

Tan Puay Siew received a Ph.D. degree in computer science from the School of Computer Engineering, Nanyang Technological University, Singapore. She is presently an Adjunct Associate Professor with the School of Computer Science and Engineering, at Nanyang Technological University. In her full-time job at A*STAR, she leads the Digital Manufacturing Division of SIMTech and ARTC. Her research interests are in the cross-field disciplines of Computer Science and Operations Research for virtual enterprise collaboration, in particular, Distributed Smart Value Chains in the era of Industry 5.0.

APPENDIX

TABLE 5 shows each attribute's accuracy by the datasets for comparison. NA in the miner denotes the cases where we could identify the key attributes in the first stage.

TABLE 5: Accuracy by datasets, miners, and k. The best accuracy is highlighted in bold.

DATASET	MINER	k	ACCURACY		
			CASE-ID	ACTIVITY	TIMESTAMP
BPIC2011	NA	1	0.00	**0.29**	**1.00**
	HM	1	0.00	0.00	**1.00**
		2	0.00	0.00	0.50
	IM	1	0.00	0.00	**1.00**
		2	0.00	0.00	0.45
BPIC2012	NA	1	0.70	0.90	**1.00**
	HM	2	0.48	**1.00**	**1.00**
	IM	2	**0.80**	**1.00**	0.71
BPIC2013 incident management	NA	1	**1.00**	0.50	**1.00**
	HM	2	0.52	0.78	0.90
	IM	2	0.76	**0.83**	0.90
BPIC2013 problem management (closed)	NA	1	0.50	0.80	**1.00**
	HM	2	**0.98**	0.82	**1.00**
	IM	2	0.76	**0.89**	**1.00**
BPIC2013 problem management (open)	NA	1	**1.00**	0.90	**1.00**
	HM	2	0.88	0.90	**1.00**
	IM	2	0.82	**0.95**	**1.00**
BPIC2015 1	NA	1	**1.00**	0.00	0.00
	HM	1	**1.00**	0.48	0.00
		2	0.96	**0.58**	0.00
	IM	1	**1.00**	0.50	0.00
		2	**1.00**	0.50	0.00
BPIC2015 2	NA	1	0.67	0.00	**1.00**
	HM	1	**0.86**	**0.43**	**0.86**
		2	0.69	0.35	0.83
	IM	1	**0.86**	**0.43**	**0.86**
		2	0.54	0.30	0.48
BPIC2015 3	NA	1	**1.00**	0.00	**1.00**
	HM	1	**1.00**	**0.28**	**1.00**
		2	0.56	0.08	0.52
	IM	1	**1.00**	0.25	**1.00**
		2	0.55	0.07	0.47
BPIC2015 4	NA	1	**1.00**	0.00	0.00
	HM	1	**1.00**	0.43	0.00
		2	0.72	**0.45**	**0.22**
	IM	1	**1.00**	0.44	0.00
		2	0.88	0.40	0.15
BPIC2015 5	HM	1	**0.90**	0.46	0.15
		2	0.48	0.41	0.00
	IM	1	**0.90**	**0.66**	**0.18**
		2	0.55	0.40	0.00
BPIC2017	NA	1	0.40	0.00	**1.00**
	HM	2	0.34	**0.11**	**1.00**
	IM	2	**0.78**	0.02	**1.00**
BPIC2018	NA	1	0.00	0.00	**1.00**
	HM	1	0.00	0.00	**1.00**
		2	**0.01**	0.00	0.73
	IM	1	0.00	0.00	**1.00**
		2	**0.01**	**0.01**	0.69
BPIC2019	NA	1	0.25	**0.62**	**1.00**
	HM	1	0.00	0.00	**1.00**
		2	0.57	0.14	**1.00**
	IM	1	0.00	0.10	**1.00**
		2	**0.69**	0.18	**1.00**
BPIC2020	NA	1	0.00	**0.33**	**1.00**
	HM	1	0.00	0.25	**1.00**
		2	0.00	0.24	0.85
	IM	1	0.00	0.25	**1.00**
		2	0.00	0.32	0.85