2020 IEEE Third International Conference on Artificial Intelligence and Knowledge Engineering (AIKE) | 978-1-7281-8708-2/20/$31.00 ©2020 IEEE | DOI: 10.1109/AIKE48582.2020.00017

Using Event Log Timing Information to Assist Process Scenario Discoveries

Zhenyu Zhang[1,3], Chunhui Guo[2], Wenyu Peng[1], Shangping Ren[1]
[1]Department of Computational Science, San Diego State University, San Diego, CA 92182, USA
[2]Department of Computer Science, California State University, Los Angeles, CA 90032, USA
[3]Department of Computer Science, University of California, Irvine, CA 92697, USA
zzhang4430@sdsu.edu, cguo4@calstatela.edu, wpeng7231@sdsu.edu, sren@sdsu.edu

***Abstract*—Event logs contain abundant information, such as activity names, time stamps, activity executors, etc. However, much of existing trace clustering research has been focused on applying activity names to assist process scenarios discovery. In addition, many existing trace clustering algorithms commonly used in the literature, such as k-means clustering approach, require prior knowledge about the number of process scenarios existed in the log, which sometimes are not known aprior. This paper presents a two-phase approach that obtains timing information from event logs and uses the information to assist process scenario discoveries without requiring any prior knowledge about process scenarios. We use five real-life event logs to compare the performance of the proposed two-phase approach for process scenario discoveries with the commonly used k-means clustering approach in terms of model's harmonic mean of the weighted average fitness and precision, i.e., the F1 score. The experiment data shows that (1) the process scenario models obtained with the additional timing information have both higher fitness and precision scores than the models obtained without the timing information; (2) the two-phase approach not only removes the need for prior information related to k, but also results in a comparable F1 score compared to the optimal k-means approach with the optimal k obtained through exhaustive search.**

I. Introduction

Over the past decade, process mining has emerged as a new research area that uses available data, such as event logs, to understand how processes are being executed in real life. Given an event log, process mining aims to extract actionable process knowledge (e.g., process models) and provides valuable insights to help better understand, monitor, and improve the current processes. In an environment where different scenarios exist, such as in healthcare [1], customer support [2], and engineering fault diagnosis [3], it becomes more critical to discover the processes that actually take place. The most important learning task in the broad field of process mining is called *process discovery*, which is concerned with the derivation of process models from event logs. Over time, a range of process discovery algorithms have been proposed, such as Alpha algorithm [4], region-based approaches [5], [6], and heuristic approach [7], to name a few. Despite the demonstrated usefulness of process discovery algorithms, these algorithms face challenges in an environment where different scenarios exist [8], [9], [10], [11]. When different scenarios are grouped into one process model not only the accuracy of the model representing the reality reduces, more importantly, the complexity of the model becomes incomprehensible and hence makes it difficult, if not impossible, to achieve the goal of better understanding, monitoring and improving the current processes.

Trace clustering techniques are often used to assist in discovering different process scenarios. Most existing trace clustering methods [9], [12], [13], [14] often contain two major steps. First, use a set of transformation rules to convert each trace in a given event log into a vector. Second, apply clustering algorithms, such as k-means [13], to the vectors and partition the vectors into different clusters, and hence partition the corresponding event log into different subsets of logs where event traces in the same subset most likely belong to the same scenario. Once an event log is partitioned into different clusters, process discovery techniques are applied to individual clusters to obtain process models for different scenarios.

Event logs often contain rich information, such as activity names, time stamps, activity executors, etc. However, our observation is that most transformation rules used in trace clustering techniques [12], [13], [9], [14] to convert an event trace to a vector only use the activity name information in the event log. We hypothesize that if additional information is added into the constructing vectors, we shall obtain process models that are better in terms of model fitness and precision criteria. We have also observed that clustering algorithms that are commonly used in trace clustering, e.g., k-means [13], [14], [9], need a prior knowledge about the number of clusters k, which sometimes may not be available. The question is can we derive it from given information, i.e., the event log?

In this paper, we present an approach that uses timing information to assist in discovering process scenarios from event logs. This approach has three steps. First, we obtain time dependent sets from a process model produced by applying an existing process discovery algorithm on the entire event logs. Second, we construct aggregated vectors that contain both activity information and timing information which is derived from the time dependent set and event time stamps. Third, we use a two-phase approach to generate subsets of event logs where event traces in the same subset are most likely under the same scenarios. The first phase of the approach is to identify the number of scenarios (k) a given event log may have. The second phase is to apply an existing distance-based clustering approach, i.e., k-means algorithm [13], [14], to partition the event log into k subsets of an event log. To evaluate the performance of the proposed approach for process scenario discoveries, we apply an existing process discovery algorithm, e.g., the *HeuristicsMiner* algorithm [15], to obtain a process model from each subset and compare the weighted average fitness, the weighted average precision, and $F1$ score against the commonly used clustering approach, i.e., the existing approach [13]. Figure. 1 depicts the workflow of our approach.

The paper is organized as follows. Section II introduces definitions used in the rest of the paper and the steps of obtaining time dependent set. We develop an approach to construct vectors by aggregating activity and timing information in Section III. Section IV presents the two-phase clustering approach to assist in discovering process scenarios. Section V evaluates the proposed approach in terms of three criteria, i.e., the weighted average fitness, the weighted average precision, and $F1$ score using real life even logs, and discusses our findings. Section VI discusses the related work and we conclude in Section VII.

II. Obtain time dependent set from event logs

In this section, we first formalize notations and definitions related to event logs that will be used in this paper, and then briefly introduce the process of obtaining the time dependent set for each activity in a process model derived from event logs.

A. Event logs

The starting point of process mining is event logs which record information about activities as they take place. We adopt the definitions similar to ones given in [16], but with additional timing information.

978-1-7281-8708-2/20/$31.00 ©2020 IEEE
DOI 10.1109/AIKE48582.2020.00017

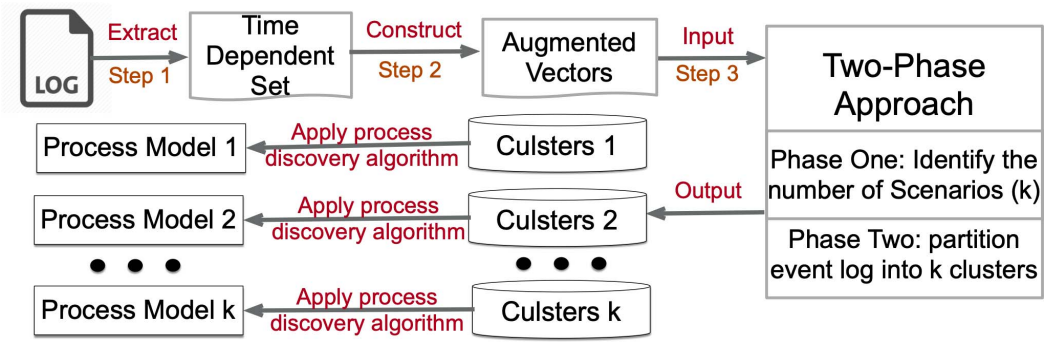

Fig. 1. Workflow of using timing information and two-phase approach for process scenario discovery

Definition 1 (Event Entry (e)). *An event entry e is a tuple (α, τ), i.e., $e = (\alpha, \tau)$, where α is activity's name and τ is the timestamp of activity α. The sets of all event activity names, activity timestamps, and events in a given event log are denoted as Ω, $\mathcal{T}$ and $\mathcal{E}$, respectively.*

Definition 2 (Attribute Function ($\mathcal{F}_\alpha$ and $\mathcal{F}_\tau$)). *The attribution functions are defined to obtain an event entry's activity name and time stamps, respectively, i.e., $\mathcal{F}_\alpha : \mathcal{E} \mapsto \Omega$ and $\mathcal{F}_\tau : \mathcal{E} \mapsto \mathcal{T}$.*

Definition 3 (Event Trace (σ)). *An event trace σ is a finite sequence of event entries $e_1, \cdots, e_n$, i.e., $\sigma = [e_1, \cdots, e_n]$, where $\mathcal{F}_\tau(e_i) < \mathcal{F}_\tau(e_{i+1})$ and $1 \leq i \leq n$.*

Definition 4 (Event Log (L)). *An event log L is a set of event traces, i.e., $L = \{\sigma_1, \cdots, \sigma_m\}$. The number of traces in the event log L is denoted as $|L|$.*

For an illustration purpose, we consider a simplified one day event log shown in Table I. This event log L has five traces, i.e., $L = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5\}$ and $|L| = 5$. The log also shows that for trace σ_1, there are four event entries, i.e., $\sigma_1 = [e_1, e_2, e_3, e_4]$. It is worth pointing out that the ordering of event entries within a trace is important, while the ordering of event entries among different traces is of no significance. The activity name and its timestamp of an event entry in an event trace are obtained by function $\mathcal{F}_\alpha$ and $\mathcal{F}_\tau$, respectively. For instance, the activity name of e_2 in trace σ_1 is $\mathcal{F}_\alpha(e_2) = B$, and its timestamp is $\mathcal{F}_\tau(e_2) =$ "10 : 24".

TABLE I
EXAMPLE OF AN EVENT LOG

Trace Identifier	Activity	Timestamp
Trace 1 (σ_1)	A	08:15
Trace 2 (σ_2)	A	08:24
Trace 3 (σ_3)	A	09:30
Trace 1 (σ_1)	B	10:24
Trace 3 (σ_3)	B	10:24
Trace 2 (σ_2)	C	10:26
Trace 1 (σ_1)	C	10:25
Trace 4 (σ_4)	A	11:45
Trace 2 (σ_2)	B	11:46
Trace 2 (σ_2)	D	12:23
Trace 5 (σ_5)	A	13:14
Trace 4 (σ_4)	C	13:17
Trace 1 (σ_1)	D	13:19
Trace 3 (σ_3)	C	14:09
Trace 3 (σ_3)	D	14:29
Trace 4 (σ_4)	B	14:43
Trace 5 (σ_5)	E	15:22
Trace 5 (σ_5)	D	15:45
Trace 4 (σ_4)	D	16:10

B. Time Dependent Set

In our previous work that aims to obtain possible timing constraints from a given event log [17], we introduced the concept of activity's *time dependent set* defined below.

Definition 5 (Time Dependent Set Θ). *Given an event log (L) and its corresponding process model obtained by any exiting process mining algorithm, the time dependent set ($\Theta(\alpha)$) of an activity α is the set containing all activities directly followed by the activity α in the process model.*

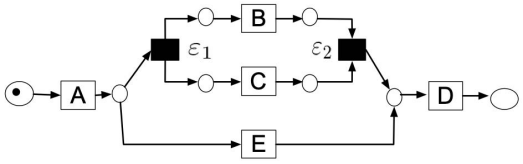

Fig. 2. A process model corresponding to the event log given in Table I

We use an example to show how an activity's time dependent set is obtained.

Example 1. *Consider the event log given in Table I and a corresponding process model shown in Fig. 2 which is derived by applying an inductive miner algorithm [18] on the given log. In the process model constructed by the inductive miner algorithm, there are activities represented in rectangles and invisible activities represented in black rectangles, such as ε_1 and ε_2, in Fig 2. Invisible activities and circles in the process model are only for routing purposes. They are produced by process mining algorithms, but not recorded in event logs.*

For activity D's time dependent set, as the invisible activity ε_2 is connected to the target activity D, we need to trace back until we find a visible activity that connects to the invisible activity ε_2, which are activity B and C. Hence, activity D's time dependent set $\Theta(D) = \{E, B, C\}$. Similarly, the time dependent sets of the activity A, B, C, and D are $\Theta(A) = \emptyset, \Theta(B) = \{A\}, \Theta(C) = \{A\}$, and $\Theta(E) = \{A\}$, respectively.

The algorithm for obtaining the time dependent set for each activity in a process model can be found in our earlier work [17].

III. CONSTRUCT AGGREGATED VECTORS WITH ACTIVITY AND TIMING INFORMATION

In this section, we present an approach to automatically construct aggregated vectors that contain both activity name from event traces and timing information derived from the time dependent set and event entry timestamps.

We first apply a similar approach used in [13] to map each event trace σ in a given event log into an *activity vector* ($\vec{\mathcal{A}}_\sigma$) which is defined below.

Definition 6 (Activity Vector ($\vec{\mathcal{A}}$)). *Given an event trace σ and an activity set Ω, the activity vector $\vec{\mathcal{A}}$ corresponding to an event trace σ is $\vec{\mathcal{A}}_\sigma = (\mathcal{N}_1(\alpha_1), \cdots, \mathcal{N}_n(\alpha_i))$, where $n = |\Omega|, \alpha_i \in \Omega$, and $\mathcal{N}(\alpha_i)$ is the number of times the activity α_i occurs in the trace σ.*

Based on the definition, for event log L given in Table I, we have the activity set $\Omega = \{A, B, C, D, E\}$, hence the size of all activity vector $|\vec{\mathcal{A}}| = 5$. Furthermore, for **Trace 1** (σ_1), its corresponding activity vector $\vec{\mathcal{A}}_{\sigma_1} = (1, 1, 1, 1, 0)$, where $\vec{\mathcal{A}}_{\sigma_1}[5] = 0$ indicates that activity E does not occur in **Trace 1**.

There is another important information contained in an event trace, i.e., the time stamp of each recorded activity. Our idea is to extend the activity vector by adding timing information related activities in a given trace. Our hypothesis is with additional activity related to timing information, we will be able to discover more accurate scenarios. Our experiments shown in Section V confirms this hypothesis.

To obtain the timing information related to each activity in a given trace, we first introduce the following definitions.

Definition 7 (Time Dependent Pair (α, β)). *Given an event log's activity set Ω, for any two activities $\alpha, \beta \in \Omega$, if $\beta \in \Theta(\alpha)$, the activities α and β are called a time dependent pair denoted as (α, β), and the set of all time dependent pairs in a given activity set Ω is denoted as $\mathcal{O} = \bigcup_{\alpha \in \Omega} \{(\alpha, \beta) | \beta \in \Theta(\alpha)\}$.*

For instance, in Example 1, activity D's time dependent set is $\Theta(D) = \{B, C, E\}$, the corresponding time dependent pairs are $(D, B), (D, C)$, and (D, E). Furthermore, the set of all time dependent pairs corresponding to the event log presented in Example 1 is $\mathcal{O} = \{(B, A), (C, A), (D, B), (D, C), (D, E), (E, A)\}$.

Definition 8 (Timing Vector ($\vec{\mathcal{T}}$)). *Given a time dependent pair set $\mathcal{O}$, the timing vector $\vec{\mathcal{T}}$, $|\vec{\mathcal{T}}| = |\mathcal{O}|$, represents the maximal time durations between two activities of a given time dependent pair $(\alpha, \beta) \in \mathcal{O}$. For a given trace $\sigma = [e_1, \cdots, e_m]$, the value of the timing vector $\vec{\mathcal{T}}_\sigma[(\alpha, \beta)]$ is defined as: $\vec{\mathcal{T}}_\sigma[(\alpha, \beta)] = \max\{\mathcal{F}_\tau(e_i) - \mathcal{F}_\tau(e_j) \mid 1 \le i, j \le m \wedge i \ne j \wedge e_i, e_j \in \sigma \wedge \mathcal{F}_\alpha(e_i) = \alpha \wedge \mathcal{F}_\alpha(e_j) = \beta\}$.*

Consider **Trace** 1(σ_1) in Table I and a time dependent pair (B, A). The trace σ_1 contains the activity B at time "10:24", the activity A's time stamp is "08:15". The time duration between the activity A's time stamp and the activity B's time stamp is 129 with minutes as the time units. Hence we have $\vec{\mathcal{T}}_{\sigma_1}[(B, A)] = 129$, indicating that activity B occurs at most 129 time units after the occurrence of activity A in σ_1. Algorithm 1 gives the detailed steps of obtaining the timing vector from an event trace.

To obtain a vector that contains both activity information and timing information in an event trace, we use weighted concatenation to aggregate the activity and timing vectors.

Definition 9 (Aggregated Vector ($\vec{\mathcal{V}}$)). *Given an event trace σ, assume the corresponding activity and timing vectors are $\vec{\mathcal{A}}_\sigma = (a_1, \cdots, a_i, \cdots, a_m)$ and $\vec{\mathcal{T}}_\sigma = (t_1, \cdots, t_j, \cdots, t_n)$, respectively, the aggregated vector $\vec{\mathcal{V}}$ corresponding to an event trace σ is $\vec{\mathcal{V}}_\sigma = (\omega a_1, \cdots, \omega a_i, \cdots, \omega a_m, \upsilon t_1, \cdots, \upsilon t_j, \cdots, \upsilon t_n)$, where $a_i = \vec{\mathcal{A}}_\sigma[i]$, $t_j = \vec{\mathcal{T}}_\sigma[j]$, and $\omega, \upsilon \in \mathcal{R}^+$.*

Consider **Trace** 1(σ_1) in Table I, assume the weighted values of activity vector and timing vector are both one, i.e., $\omega = 1.0$ and $\upsilon = 1.0$, the aggregated vector corresponding to σ_1 is $\vec{\mathcal{V}}_{\sigma_1} = (1, 1, 1, 1, 0, 129, 130, 175, 174, 0, 0)$. Table II gives the vector value for the other four traces. Note that, the weighted values provide a way to emphasize which perspective should have more impact on the outcome of discovery process scenarios.

Algorithm 1 CONSTRUCT TIMING VECTOR

Input: An event trace $\sigma = \{e_1, \cdots, e_n\}$ and the set of all time dependent pairs $\mathcal{O} = \bigcup_{\alpha \in \Omega} \{(\alpha, \beta) | \beta \in \Theta(\alpha)\}$

Output: The corresponding timing vector $\vec{\mathcal{T}}_\sigma = (\Gamma(\alpha_1, \beta_1), \cdots, \Gamma(\alpha_i, \beta_j))$, where the $\Gamma(\alpha_i, \beta_j)$ indicates the maximal time duration between two activities of a given time dependent pair $(\alpha_i, \beta_j) \in \mathcal{O}$

```
for i = 1 to n do
    j = i − 1
    while j ≠ 0 do
        if (e_i, e_j) ∈ O then
            if F_τ(e_i) − F_τ(e_j) > Γ(F_α(e_i), F_α(e_j)) then
                Γ(F_α(e_i), F_α(e_j)) = F_τ(e_i) − F_τ(e_j)
            end if
        end if
        j = j − 1
    end while
end for
```

IV. TWO-PHASE PROCESS SCENARIO DISCOVERY APPROACH USING AGGREGATED VECTORS

In this section, we develop a two-phase approach to discover the process scenarios from a given event log using aggregated vectors. As mentioned in Section I, for a given event log, often times, we do not know aprior how many different scenarios are embedded in the log. Hence, the first phase of our two-phase approach is to identify the number of possible scenarios k in a given log. The second phase is to apply an existing clustering approach, such as the k-means algorithm [13], to partition the event log into k clusters based on the aggregated vectors so that existing process discovery algorithms can be applied on the clusters of sub-logs to obtain different process scenarios in terms of process models.

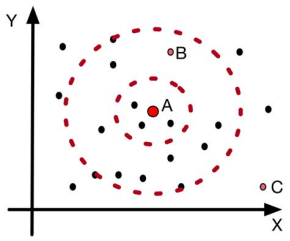

Fig. 3. Illustration of the strategy about the first phase

We use Fig. 3 to illustrate the basic idea used in the first phase to obtain the number of possible scenarios k. For simplicity, assume the cardinality of event trace's aggregated vector $|\vec{\mathcal{V}}| = 2$, therefore, each aggregated vector (a trace) can be mapped to a point in a two-dimensional domain shown in Fig. 3. Consider the centroid point A, we can define two radius circles, the inner circle, and the outer circle. The points within the inner circle of A have short distance from A, or can be considered 'similar' to A, while the points outside the outer circle (such as C) are considered far away or different from A. The radius are the thresholds for defining if two points are similar or different. The points outside the inner circle but within the outer cycle, such as point B, may or may not have similar characteristics to A. For these points, we repeat the process of choosing a centroid point and defining similar and different points from the new centroid points, until all points are categorized. Then

TABLE II
AGGREGATED VECTORS TRANSFERRED FROM THE EVENT LOG PRESENTED IN TABLE I

	Activity Information					Timing Information					
	$\mathcal{N}(A)$	$\mathcal{N}(B)$	$\mathcal{N}(C)$	$\mathcal{N}(D)$	$\mathcal{N}(E)$	(B,A)	(C,A)	(D,B)	(D,C)	(D,E)	(E,A)
$\vec{\mathcal{V}}_{\sigma_1}$	1	1	1	1	0	129	130	175	174	0	0
$\vec{\mathcal{V}}_{\sigma_2}$	1	1	1	1	0	202	122	37	117	0	0
$\vec{\mathcal{V}}_{\sigma_3}$	1	1	1	1	0	54	279	245	20	0	0
$\vec{\mathcal{V}}_{\sigma_4}$	1	1	1	1	0	178	92	87	183	0	0
$\vec{\mathcal{V}}_{\sigma_5}$	1	0	0	1	1	0	0	0	0	23	151

the number of centroids selected during the process is the k. There are many existing techniques in the literature to identify the radius of inner and outer circles, i.e., the distance thresholds, such as the cross validation approach [19]. Sometimes, the distance threshold can also be set by domain experts [12].In this work, we adopt the upper quartile and the lower quartile as the two distance thresholds.

Let $d(\vec{\mathcal{V}}_i, \vec{\mathcal{V}}_j)$ denote *Euclidean* distances between aggregated vectors $\vec{\mathcal{V}}_i$, $\vec{\mathcal{V}}_j$, Algorithm 2 shows the procedure to find the number of the scenarios from aggregated vectors. The time complexity of the algorithm is $O(N^2)$, where N is the number of event traces in the event log L. We use Example 2 to illustrate Algorithm 2.

Algorithm 2 ALGORITHM OF THE FIRST PHASE

Input: A set of aggregated vectors $\Pi = \{\vec{\mathcal{V}}_1, \cdots, \vec{\mathcal{V}}_{|L|}\}$ and a set of *Euclidean* distances $\mathcal{M} = \{d(\vec{\mathcal{V}}_i, \vec{\mathcal{V}}_j) | \vec{\mathcal{V}}_i, \vec{\mathcal{V}}_j \in \Pi\}$
Output: The number of clusters (k) and a set of centroids $\mathcal{C}$ for the k clusters' aggregated vectors

1: Let δ_{min} and δ_{max} be the first quartile and the third quartile of the elements in $\mathcal{M}$, respectively.
2: $k \leftarrow 0$ and $\mathcal{C} \leftarrow \emptyset$
3: **while** $\Pi \neq \emptyset$ **do**
4: Select an arbitrary aggregated vector $\vec{\mathcal{V}}_i \in \Pi$
5: $\Pi \leftarrow \Pi - \{\vec{\mathcal{V}}_i\}$
6: $\Delta \leftarrow \{\vec{\mathcal{V}}_i\}$
7: $k \leftarrow k + 1$
8: **for each** $\vec{\mathcal{V}}_j \in \Pi$ **do**
9: **if** $d(\vec{\mathcal{V}}_i, \vec{\mathcal{V}}_j) \leq \delta_{max}$ **then**
10: $\Delta \leftarrow \Delta \cup \{\vec{\mathcal{V}}_j\}$
11: **end if**
12: **if** $d(\vec{\mathcal{V}}_i, \vec{\mathcal{V}}_j) \leq \delta_{min}$ **then**
13: $\Pi \leftarrow \Pi - \{\vec{\mathcal{V}}_j\}$
14: **end if**
15: **end for**
16: $\vec{c} \leftarrow$ calculate the initial centroid of the aggregated vectors in the set Δ
17: $\mathcal{C} \cup \{\vec{c}\}$
18: **end while**
19: **return** k and $\mathcal{C}$

Example 2. *Consider the set of aggregated vectors* $\Pi = \{\vec{\mathcal{V}}_1, \vec{\mathcal{V}}_2, \vec{\mathcal{V}}_3, \vec{\mathcal{V}}_4, \vec{\mathcal{V}}_5\}$ *given in the Table II, the set of Euclidean distances of two aggregated vectors in L is* $\mathcal{M} = \{(\vec{\mathcal{V}}_1, \vec{\mathcal{V}}_2) = 166.39, (\vec{\mathcal{V}}_1, \vec{\mathcal{V}}_3) = 237.58, (\vec{\mathcal{V}}_1, \vec{\mathcal{V}}_4) = 108.03, (\vec{\mathcal{V}}_1, \vec{\mathcal{V}}_5) = 343.18, (\vec{\mathcal{V}}_2, \vec{\mathcal{V}}_3) = 315, (\vec{\mathcal{V}}_2, \vec{\mathcal{V}}_4) = 91.28, (\vec{\mathcal{V}}_2, \vec{\mathcal{V}}_5) = 306.72, (\vec{\mathcal{V}}_3, \vec{\mathcal{V}}_4) = 299.03, (\vec{\mathcal{V}}_3, \vec{\mathcal{V}}_5) = 405.6, (\vec{\mathcal{V}}_4, \vec{\mathcal{V}}_5) = 323.32\}$, *and the first quartile and the third quartile of the elements in* $\mathcal{M}$ *is* $\delta_{min} = 5$ *and* $\delta_{max} = 10$, *respectively.*

According to Line 4-6 of algorithm, we select an aggregated vector $\vec{\mathcal{V}}_1$, *remove it from the set* Π, *and put it into the set* Δ. *Based on the line 7, the number of the scenario becomes one, i.e.* $k = 1$. *We apply Line 8-15 of algorithm for every aggregated vector in the the set* Π. *For the aggregated vector* $\vec{\mathcal{V}}_2$, *since the distance between aggregated vectors* $\vec{\mathcal{V}}_1$ *and* $\vec{\mathcal{V}}_2$, *i.e.* $d(\vec{\mathcal{V}}_1, \vec{\mathcal{V}}_2)$, *is* 166.39 *and less than* δ_{min}, *we remove the aggregated vector* $\vec{\mathcal{V}}_2$ *from the set* Π *and add* $\vec{\mathcal{V}}_2$ *into the set* Δ. *Hence, the set* Δ *is updated to* $\{\vec{\mathcal{V}}_1, \vec{\mathcal{V}}_2\}$, *and* Π *becomes* $\{\vec{\mathcal{V}}_3, \vec{\mathcal{V}}_4, \vec{\mathcal{V}}_5\}$. *For the aggregated vector* $\vec{\mathcal{V}}_3$, *the distance between aggregated vectors* $\vec{\mathcal{V}}_1$ *and* $\vec{\mathcal{V}}_3$, *i.e.* $d(\vec{\mathcal{V}}_1, \vec{\mathcal{V}}_3)$, *is* 237.58 *and less than* δ_{max}, *we add* $\vec{\mathcal{V}}_3$ *into set* Δ. *Because* $d(\vec{\mathcal{V}}_1, \vec{\mathcal{V}}_3) > \delta_{min}$, *we keep* $\vec{\mathcal{V}}_3$ *in the set* Π. *After applying the procedure to the other two aggregated vectors, the* $\Delta = \{\vec{\mathcal{V}}_1, \vec{\mathcal{V}}_2, \vec{\mathcal{V}}_3, \vec{\mathcal{V}}_4\}$, *and the* $\Pi = \{\vec{\mathcal{V}}_3, \vec{\mathcal{V}}_5\}$. *We have the initial centroid of the aggregated vectors in the set* Δ *is* $(1, 1, 1, 1, 0, -790, -87, -327, 0, 0, 0)$.

We repeat the steps until Π *becomes empty, and have* $k = 3$ *and the initial centroids are* $(1, 1, 1, 1, 0, -790, -87, -327, 0, 0, 0)$, $(1, 1, 1, 1, 0, 54, 279, 245, 20, 0, 0)$, *and* $(1, 0, 0, 1, 1, 0, 0, 0, 0, 23, 151)$.

In the second phase, we take the number of scenarios k and the initial centroids produced by the first phase as input and apply k-means algorithm [20] to partition the event log into k clusters, i.e., scenarios. Finally, the existing process discovery algorithms can be applied on the subset of logs to obtain different process scenarios in terms of process models.

Next section, we apply the two-phase approach using aggregated vectors on different sets of real-life event logs and compare the two-phase approach with existing commonly used approaches in terms of the quality of scenario models discovered.

V. EXPERIMENTAL EVALUATION

In this section, we experimentally evaluate the proposed approach on five real-life event logs [21], [22], [23], [24], [25].The characteristics of these event logs are summarized in Table III. The objectives of the evaluations are twofolds, i.e., first, evaluate if having timing information can improve the quality of process scenario models discovered from event logs. To do so, we choose an existing heuristic mining algorithm [15], and apply it on the same event logs but with and without time information. Second, compare the number of scenarios and the scenario models obtained by the two-phase approach with k-means approach where k is exhaustively searched. Before the evaluation, we first define our evaluation criteria.

TABLE III
CHARACTERISTICS OF THE EVENT LOGS USED FOR THE EVALUATION

DataSet	Number of Traces	Number of Event Entry	Average number of Event Entry Per Trace
BPIC2013 [21]	819	2351	2.871
BPIC2020 [22]	10500	56,437	5.374
Hospital Billing [23]	100000	451359	4.514
Review Process [24]	10000	236360	23.636
Road Traffic [25]	150370	561470	3.734

A. Performance Metrics

The quality of a process model is evaluated by two criteria, i.e., (1) fitness f, and (2) precision p. *Fitness* measures how well a model can reproduce the process behavior contained in a log, and the *precision*

measures the degree to which the behavior made possible by a model is found in a log [26].The higher fitness value and precision value, the better quality of the process model. We use the pm4py [27] library to calculate the fitness value and precision value.

As we may have multiple scenarios in a given event log, we need the weighted average fitness f^W and precision p^W to evaluate the quality of the multiple scenario process models. Similar to the approach in [28], the weighted average fitness and precision are calculated as follows:

$$f^W = \frac{\sum_{i=1}^{k} n_i \times f_i}{|L|}$$

$$p^W = \frac{\sum_{i=1}^{k} n_i \times p_i}{|L|}$$

where k is the number of subsets of logs representing the scenarios, n_i is the number of event traces in the ith subset of a given event log, and f_i and p_i are the fitness value and precision value of the process model derived from of subsets of logs, respectively.

The fitness and precision are two aspects of a process model which may not always be consistent. The $F1$ score [29] is defined as the harmonic mean of the weighted average fitness f^W and precision p^W, i.e. $F1 = \frac{2 \times f^W \times p^W}{f^W + p^W}$. We will also use the $F1$ score as an evaluation criteria.

B. Process Model Discovery Performance Improvement with Timing Information

This set of experiments is to evaluate whether adding the timing information in constructing the vectors can improve process scenario discovery performance. More specifically, given an event log, we apply the proposed approach presented in Section III to construct the aggregated vectors that contain both activity information and timing information. For the same event log, we also apply the heuristic mining approach presented in [13] to obtain the activity vectors that contain only the activity information. For both aggregated vectors and activity vectors, we apply the k-means algorithm to partition the event logs into k clusters, where k is from 2 to 5, and discover a process model from each subset of logs using the heuristic mining algorithm presented in [7].

We use the pm4py tool [27] with the following settings: (1) process mining algorithm is heuristic mining; (2) the dependency threshold of the algorithm is 0.9; (3) the minimum number of occurrences of an activity is 1; (4) the minimum number of occurrences of an edge is 1; and (5) the thresholds for the loops of length is 2. The weighted average fitness, the weighted average precision, and the $F1$ score of the process models are depicted in Table IV, Table V, and Table VI, respectively.

From the experiment results, it is clear that when we add timing information into the vector, we are able to achieve higher weighted average fitness, weighted average precision, and $F1$ score, under different k and with different real-life event logs.

For each event log, we also calculate the average improvement percentage of different k values which is shown in Table VII. We observe that the improvements of the weighted average precision is better than the weighted average fitness in general. The reason is that the weighted average fitness resulted from the approach in [13] is closer to 1 than the weighted average precision, which indicates the room for fitness improvement is relatively small.

C. Two-phase Scenario Discovery Vs Exhaustive Search for k with k-means clustering

The second set of experiments compare the performance of the proposed two-phase scenario discovery approach with the k-means approach. For both the two-phase scenario discovery approach and the optimal k-means solution, we use the aggregated vectors when partitioning the event log.

To obtain the optimal k-means solution, we use the brute-force approach as follow. We apply the k-means algorithm to partition the event log to a set of sub logs with the k starting from 2 with incremental step of 1 until one of the resulted subsetsbecomes empty. For each resulted cluster set, we apply the heuristic mining algorithm to generate the process models, and calculate the $F1$ score, and the average $F1$ of all resulted cluster sets from k-means approach when k varies. Both the largest and the average $F1$ scores are compared with the value obtained with the proposed solution. The results are depicted in Fig. 4.

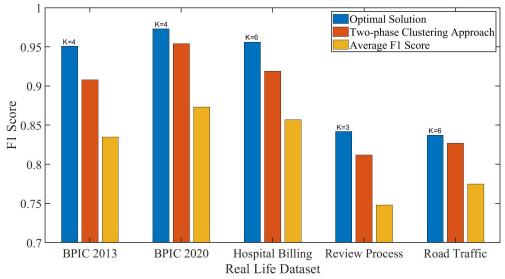

Fig. 4. $F1$ scores of two-phase clustering approach, optimal k-means approach, and average $F1$ score of k-means approach

As shown in Fig. 4, the proposed two-phase scenario discovery approach outperforms the k-means approach on average cases. More specifically, the $F1$ score of the two-phase approach is 7.73%, 8.21%, 7.15%, 7.46%, and 8.10 % higher than the average $F1$ score for *BPIC 2013*, *BPIC2020*, *Hospital Billing*, *Review Process*, and *Road Traffic* event logs, respectively. Compared with the optimal k-means solution, the two-phase approach results in lower $f1$ score, but the difference is less than 5%.

D. Internal consistency Improvement with Timing Information

VI. Related Work

Over the past decade, a range of effective process discovery algorithms have been proposed [30] in process mining field. Despite the demonstrated usefulness of process discovery algorithms, these algorithms face challenges in an environment where different scenarios exist [8], [9], [10], [11]. When different scenarios are grouped into one process model, the complexity of the model becomes incomprehensible. The work in [33] is the first study which applies the trace clustering techniques to assist in discovering the process scenarios in order to obtain more accurate and interpretable process models from event logs. This study proposed an approach to cluster the event traces based on the structural patterns frequently occurring in the event log, and then partition the corresponding event log into different subsets of logs where event traces in the same subset most likely belong to the same scenario. Once the event logs are partitioned into different clusters, process discovery techniques can be applied on the clusters to obtain process models for different scenarios.

The most straightforward idea for clustering traces methods relies on traditional data clustering techniques [31]. Greco et al. [12] were pioneers in studying the clustering of log traces within the process mining domain. They use a vector space model considering the activities to cluster the traces in an event log with the purpose of discovering more simple process models for the subgroups. The authors propose the use of disjunctive workflow schemas (DWS) for discovering process models. The underlying clustering methodology is k-means clustering. Song et al. [13] proposed an approach to construct a vector space model for traces in an event log. In [13], the author allows for different kinds of attributes, e.g., activity name, executor, to determine the vector associated with each event trace,

TABLE IV
WEIGHTED AVERAGE FITNESS

	k=2		k=3		k=4		k=5	
	Without Timing Information	With Timing Information	Without Timing Information	With Timing Information	Without Timing Information	With Timing Information	Without Timing Information	With Timing Information
BPIC2013	0.951	0.998	0.952	0.961	0.970	0.972	0.979	0.981
BPIC2020	0.985	0.998	0.986	0.998	0.986	0.998	0.989	0.998
Hospital Billing	0.785	0.998	0.973	0.995	0.945	0.957	0.945	0.997
Review Process	0.898	0.956	0.915	0.957	0.925	0.963	0.939	0.962
Road Traffic	0.768	0.976	0.823	0.976	0.963	0.979	0.969	0.975

TABLE V
WEIGHTED AVERAGE PRECISION

	k=2		k=3		k=4		k=5	
	Without Timing Information	With Timing Information	Without Timing Information	With Timing Information	Without Timing Information	With Timing Information	Without Timing Information	With Timing Information
BPIC2013	0.816	0.899	0.721	0.857	0.675	0.804	0.651	0.759
BPIC2020	0.943	0.953	0.876	0.941	0.876	0.907	0.989	0.996
Hospital Billing	0.883	0.917	0.792	0.890	0.870	0.907	0.863	0.944
Review Process	0.530	0.754	0.431	0.571	0.475	0.551	0.492	0.646
Road Traffic	0.640	0.729	0.606	0.729	0.561	0.719	0.561	0.708

TABLE VI
$F1$ SCORE

	k=2		k=3		k=4		k=5	
	Without Timing Information	With Timing Information	Without Timing Information	With Timing Information	Without Timing Information	With Timing Information	Without Timing Information	With Timing Information
BPIC2013	0.879	0.946	0.821	0.906	0.796	0.880	0.782	0.855
BPIC2020	0.963	0.975	0.927	0.969	0.927	0.950	0.940	0.964
Hospital Billing	0.831	0.956	0.873	0.939	0.906	0.948	0.902	0.959
Review Process	0.667	0.843	0.585	0.716	0.627	0.701	0.646	0.733
Road Traffic	0.698	0.835	0.688	0.837	0.709	0.829	0.709	0.819

TABLE VII
AVERAGE IMPROVEMENT PERCENTAGE

	Weighted Average Fitness	Weighted Average Precision	F1 Score
BPIC2013	2.02%	16.00%	9.53%
BPIC2020	1.24%	4.09%	2.71%
Hospital Billing	10.18%	8.06%	9.06%
Review Process	5.09%	30.30%	20.12%
Road Traffic	15.84%	20.81%	18.73%

and then provide four clustering techniques for trace clustering. In [9], [14], they extend the existing trace clustering techniques by improving the way in which control-flow context information is taken into account. The control-flow context information refers to the execution pattern of activities in the event log. Jagadeesh et al. [9] propose a generic edit distance technique based on the Levenshtein distance. The approach relies on substitution, insertion, and deletion costs to partition the event log into a set of sub logs. Bose et al. [14] propose a refinement of the technique by using conserved patterns, which are *Maximal*, *Super Maximal*, and *Near Super Maximal Repeats*. However, their implementation applies hierarchical clustering instead of k-means.

Event logs contain abundant information, such as activity names, time stamps, activity executors, etc. The most existing work focuses on applying activity names information or control-flow context information to assist process scenarios discovery. Unfortunately, not much work is done in the area of clustering traces involving timing information recorded in event logs. One exception is Appice's work [32]. In [32], the author considers that traces of an event log can be represented in multiple trace profiles derived by accounting for several perspectives such as activity, control flow, organization, and timestamp of activities. Different from Appice's work, we consider the timing information which is derived from the time dependent set and event time stamps.

VII. CONCLUSION

In this paper, we present a two-phase approach that obtains timing information from event logs and uses the information to assist process scenario discoveries. This approach does not require any prior knowledge about process scenarios. The experiments on real-life even logs show that: (1) the process scenario models obtained with the additional timing information have higher fitness and precision scores than the models obtained without the timing information; (2) the two-phase approach results in higher $F1$ scores on average compared to the k-means approach, and less than 5% lower $F1$ score compared with the optimal k obtained through exhaustive search. These results lead to the conclusions that timing information can improve process scenario discovery performance, and the proposed two-phase approach can discovers different process scenarios more effectively.

From the experiment results, we also notice that the scenario discovery process is 'mechanical' in the sense that it does not have domain experts in the loop neither utilize domain experts' knowledge in the process. In the future work, we will study how domain knowledge may be represented and also utilized in scenario discoveries.

ACKNOWLEDGEMENT

The work is supported in part by NSF CNS 1842710.

REFERENCES

[1] Ronny S Mans, Wil MP Van der Aalst, and Rob JB Vanwersch. *Process mining in healthcare: evaluating and exploiting operational healthcare processes*. Springer, 2015.

[2] Maikel Leemans and Wil MP van der Aalst. Process mining in software systems: Discovering real-life business transactions and process models from distributed systems. In *2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pages 44–53. IEEE, 2015.

[3] Jonathan E Cook and Alexander L Wolf. Discovering models of software processes from event-based data. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 7(3):215–249, 1998.

[4] 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.

[5] Marc Solé and Josep Carmona. Process mining from a basis of state regions. In *International Conference on Applications and Theory of Petri Nets*, pages 226–245. Springer, 2010.

[6] Robin Bergenthum, Jörg Desel, Robert Lorenz, and Sebastian Mauser. Process mining based on regions of languages. In *International Conference on Business Process Management*, pages 375–383. Springer, 2007.

[7] AJMM Weijters, Wil MP van Der Aalst, and AK Alves De Medeiros. Process mining with the heuristics miner-algorithm. *Technische Universiteit Eindhoven, Tech. Rep. WP*, 166:1–34, 2006.

[8] Stijn Goedertier, Jochen De Weerdt, David Martens, Jan Vanthienen, and Bart Baesens. Process discovery in event logs: An application in the telecom industry. *Applied Soft Computing*, 11(2):1697–1710, 2011.

[9] RP Jagadeesh Chandra Bose and Wil MP Van der Aalst. Context aware trace clustering: Towards improving process mining results. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, pages 401–412. SIAM, 2009.

[10] Gabriel M Veiga and Diogo R Ferreira. Understanding spaghetti models with sequence clustering for prom. In *International Conference on Business Process Management*, pages 92–103. Springer, 2009.

[11] Christian W Günther. Process mining in flexible environments. 2009.

[12] Gianluigi Greco, Antonella Guzzo, Luigi Pontieri, and Domenico Sacca. Discovering expressive process models by clustering log traces. *IEEE Transactions on knowledge and data engineering*, 18(8):1010–1027, 2006.

[13] Minseok Song, Christian W Günther, and Wil MP Van der Aalst. Trace clustering in process mining. In *International conference on business process management*, pages 109–120. Springer, 2008.

[14] RP Jagadeesh Chandra Bose and Wil MP van der Aalst. Trace clustering based on conserved patterns: Towards achieving better process models. In *International Conference on Business Process Management*, pages 170–181. Springer, 2009.

[15] Anton JMM Weijters and Wil MP Van der Aalst. Rediscovering workflow models from event-based data using little thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162, 2003.

[25] Road traffic fine management process. https://data.4tu.nl/repository/uuid:270fd440-1057-4fb9-89a9-b699b47990f5.

[16] Wil Van Der Aalst. *Process mining: discovery, conformance and enhancement of business processes*, volume 2. Springer, 2011.

[17] Zhenyu Zhang, Chunhui Guo, and Shangping Ren. Mining timing constraints from event logs for process model. In *2020 IEEE 44th Annual Computer Software and Applications Conference (COMPSAC)*. IEEE, 2020.

[18] Alejandro Bogarín Vega, Rebeca Cerezo Menéndez, and Cristobal Romero. Discovering learning processes using inductive miner: A case study with learning management systems (lmss). *Psicothema*, 2018.

[19] Michael W Browne. Cross-validation methods. *Journal of mathematical psychology*, 44(1):108–132, 2000.

[20] Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE transactions on pattern analysis and machine intelligence*, 24(7):881–892, 2002.

[21] Bpi challenge 2013. https://data.4tu.nl/repository/uuid:3537c19d-6c64-4b1d-815d-915ab0e479da.

[22] Bpi challenge 2020. https://data.4tu.nl/repository/uuid:3f422315-ed9d-4882-891f-e180b5b4feb5.

[23] Hospital billing - event log. https://data.4tu.nl/repository/uuid:76c46b83-c930-4798-a1c9-4be94dfeb741.

[24] Synthetic event logs - review example. https://data.4tu.nl/repository/uuid:da6aafef-5a86-4769-acf3-04e8ae5ab4fe.

[26] Raffaele Conforti, Marcello La Rosa, and Arthur HM ter Hofstede. Filtering out infrequent behavior from business process event logs. *IEEE Transactions on Knowledge and Data Engineering*, 29(2):300–314, 2016.

[27] *State-of-the-art-process mining in Python*, 2020. https://pm4py.fit.fraunhofer.de/.

[28] Jochen De Weerdt, Seppe Vanden Broucke, Jan Vanthienen, and Bart Baesens. Active trace clustering for improved process discovery. *IEEE Transactions on Knowledge and Data Engineering*, 25(12):2708–2720, 2013.

[29] Wil Van Der Aalst. Data science in action. In *Process mining*, pages 3–23. Springer, 2016.

[30] Adriano Augusto, Raffaele Conforti, Marlon Dumas, Marcello La Rosa, Fabrizio Maria Maggi, Andrea Marrella, Massimo Mecella, and Allar Soo. Automated discovery of process models from event logs: Review and benchmark. *IEEE Transactions on Knowledge and Data Engineering*, 31(4):686–705, 2018.

[31] Anil K Jain and Richard C Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[32] Annalisa Appice and Donato Malerba. A co-training strategy for multiple view clustering in process mining. *IEEE transactions on services computing*, 9(6):832–845, 2015.

[33] Francesco Folino, Gianluigi Greco, Antonella Guzzo, and Luigi Pontieri. Mining usage scenarios in business processes: Outlier-aware discovery and run-time prediction. *Data & Knowledge Engineering*, 70(12):1005–1029, 2011.

[34] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to data mining*. Pearson Education India, 2016.