Journal of Systems Architecture 125 (2022) 102435

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Process scenario discovery from event logs based on activity and timing information

Zhenyu Zhang [a,b,*], Caleb Johnson [a], Nalini Venkatasubramanian [b], Shangping Ren [a]

[a] Department of Computer Science, San Diego State University, San Diego, CA 92182, United States of America
[b] Department of Computer Science, University of California Irvine, Irvine, CA 92697, United States of America

ARTICLE INFO

Keywords:
Process mining
Process scenario discovery
Event logs
Wastewater treatment
Cyber–physical systems

ABSTRACT

Wastewater treatment processes are inherently dynamic due to the large variations in influent wastewater flow rate, weather conditions, concentration,and composition. The utilization of trace clustering techniques in process mining research field is an excellent way to analyze both the execution and confirm the compliance of wastewater treatment processes. However, much of existing trace clustering research has been focused on applying activity names to assist process scenarios discovery without considering other information in event logs. In addition, many existing 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 an approach that uses timing information to assist in discovering process scenarios from event logs in wastewater treatment processes without requiring any prior knowledge about process scenarios. A real wastewater treatment process provided by a domain expert is used as a case study to investigate the effectiveness and validity of the approach. We also use five real-life event logs to compare the performance of the proposed 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 proposed approach is able to discover the process scenarios from event logs in wastewater treatment domain; (2) 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.

1. Introduction

Wastewater treatment processes are inherently dynamic due to the large variations in influent wastewater flow rate, weather conditions, concentration, and composition. Due to these characteristics, the study of wastewater treatment processes requires a specific and non-generic approach. Process mining has emerged as a new research field in the past decade that uses data, generally in the form of event logs, to understand and improve real-world processes. Specifically in the domain of wastewater treatment, where different scenarios exist, the utilization of process mining is an excellent way to analyze both the execution and confirm the compliance of wastewater treatment processes.

The most important learning task within the field of process mining is 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 [1], region-based approaches [2,3], and heuristic approach [4], to name a few. Despite the demonstrated usefulness of process discovery algorithms, they face challenges in an environment where different scenarios exist [5–8]. When different scenarios are grouped into one process model not only the accuracy of the model representing the reality reduces, but, more importantly, the complexity of the model becomes incomprehensible. This results in it being 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 [6,9–11] 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 [10], to the vectors and partition them into different clusters. This results in partitioning 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

* Corresponding author at: Department of Computer Science, San Diego State University, San Diego, CA 92182, United States of America.
E-mail addresses: zzhang4430@sdsu.edu (Z. Zhang), cjohnson6200@sdsu.edu (C. Johnson), nalini@ics.uci.edu (N. Venkatasubramanian), sren@sdsu.edu (S. Ren).

https://doi.org/10.1016/j.sysarc.2022.102435
Received 21 October 2021; Received in revised form 31 January 2022; Accepted 6 February 2022
Available online 17 February 2022

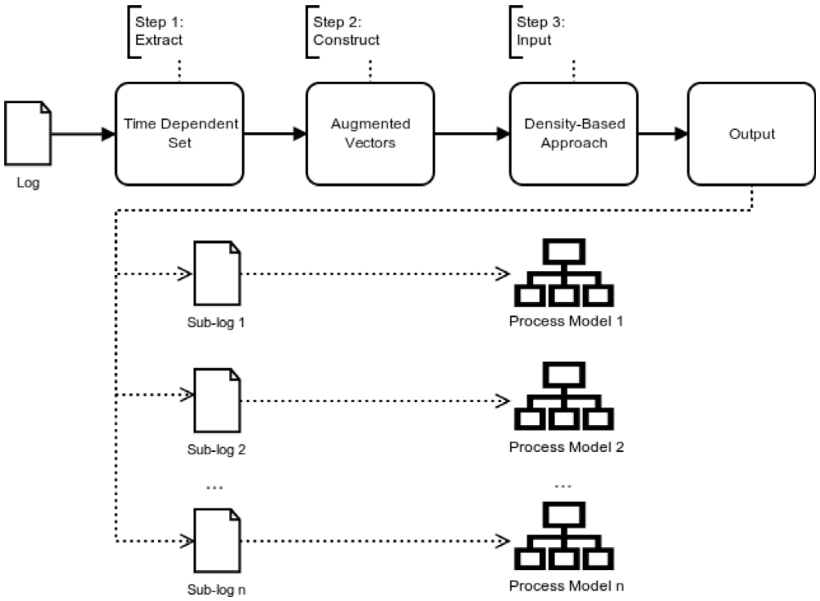

Fig. 1. Workflow of using timing information and density-based approach for process scenario discovery in wastewater treatment process.

techniques are applied to individual clusters to obtain process models for different scenarios.

Event logs often contain rich information, such as activity names, timestamps, activity executors, etc. However, our observation is that most transformation rules used in trace clustering techniques [6,9–11] 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 more accurately reflect model fitness and precision criteria. We have also observed that the commonly used trace clustering algorithms, e.g., k-means [6,10,11], need a priori knowledge about the number of scenarios k, which is unfortunately not available for wastewater treatment application. The question is, do we have an approach to offer a good solution without the requirements of domain knowledge related to the number of scenarios? The last observation is that the shape of all clusters found by commonly used trace clustering techniques [6,10,11] is convex which is very restrictive. Due to the characteristics of wastewater treatment processes, we should propose an approach for discovery of clusters with arbitrary shape.

In this paper, we present an approach that uses timing information to assist in discovering process scenarios from event logs in wastewater treatment processes. This approach has three steps. First, we obtain time dependent sets from a process model produced by applying an existing process discovery algorithm on 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 apply a density-based approach to generate subsets of event logs where event traces in the same subset are most likely under the same scenarios. A real wastewater treatment process provided by a domain expert is used as a case study to investigate the effectiveness and validity of the approach. To evaluate the performance of the proposed approach for process scenario discoveries, we apply an existing process discovery algorithm, e.g., the *HeuristicsMiner* algorithm [12], to obtain a process model from each subset. We then compare the weighted average fitness, the weighted average precision, and $F1$ score against the commonly used clustering approach, i.e., the existing approach [10]. Fig. 1 depicts the workflow of our approach.

The paper is organized as follows. Section 2 introduces definitions and notations used in the rest of the paper. We develop an approach to assist in discovering process scenarios in wastewater treatment process in Section 3. Section 4 investigates the effectiveness of the approach using a real wastewater treatment process and 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 5 discusses the related work and we conclude in Section 6.

2. Definitions and notations

In this section, we first formalize notations and definitions related to event logs used in this paper, and then introduce the process model in terms of a Petri net . Some of the definitions are cited here for self-containment.

2.1. Event logs

The starting point for process mining is an event log. An event log records information about activities as they take place. For this paper, we adopt definitions that are similar to the ones given in [13]. In particular, for a given activity set Ω, an event entry e in an event log records an activity happening within the operation of a process. An event trace σ is a finite sequence of event entries that are ordered by their occurrence time. An event log L consists of a set of event traces. The formal definitions of an event entry, an event trace, and an event log are given below.

Definition 1 (*Event Entry*). Given an activity set Ω, an event entry e is a tuple (α, τ), where $\alpha \in \Omega$ is an activity and τ is the timestamp of α. Labeling functions act: $e \mapsto \alpha$ is used to get the activity of the event entry. □

Definition 2 (*Event Trace*). An event trace σ is a finite sequence of event entries $[e_1, \ldots, e_i, \ldots, e_n]$ where $e_i = (\alpha_i, \tau_i)$. □

Definition 3 (*Event Log*). An event log L is a set of event traces $\{\sigma\}$. $|L|$ denotes the cardinality of L. □

For illustrative purposes, we consider a simplified event log as shown in Table 1. This event log L contains information about five traces, i.e., $L = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5\}$. Note that the ordering of activities within a trace is relevant, while the ordering of activities among different traces is of no importance. The log shows that for **trace** 1, the activities A, B, C, D were executed, i.e., $\sigma_1 = [e_1, e_2, e_3, e_4]$ where

Table 1
An example of event logs.

Trace identifier	Activity	Timestamp
Trace 1	A	08:15
Trace 2	A	08:24
Trace 3	A	09:30
Trace 1	B	10:24
Trace 3	B	10:24
Trace 2	C	10:26
Trace 1	C	10:25
Trace 4	A	11:45
Trace 2	B	11:46
Trace 2	D	12:23
Trace 5	A	13:14
Trace 4	C	13:17
Trace 1	D	13:19
Trace 6	A	13:39
Trace 3	C	14:09
Trace 6	B	14:19
Trace 3	D	14:29
Trace 4	B	14:43
Trace 5	E	15:22
Trace 6	C	15:29
Trace 5	D	15:45
Trace 4	D	16:10
Trace 6	D	16:43

$e_1 = (A, \text{“08 : 15”})$, $e_2 = (B, \text{“10 : 24”})$, $e_3 = (C, \text{“10 : 25”})$, and $e_4 = (D, \text{“13 : 19”})$. Due to the page limit, we omit the representations of the **trace** 2, 3, 4, 5, and 6 which are similar to the representation of **trace** 1. Each trace starts with the execution of activity A and ends with activity D. We also observe that if activity C is executed within a trace, activity B is also executed. However, for some traces, activity C is executed before activity B, while for some others, it is the other way around. The activity of event entries in a event trace is obtained a by labeling function. For instance, the activity of e_1 in **trace** 1 is A, i.e., $act(e_1) = A$.

2.2. Petri net

The majority of process mining algorithms [3,14,15] use a Petri net [2] to represent process models.

Definition 4 (*Petri net [16]*)**.** A Petri net N is a tuple (P, T, F), where

- P is a finite set of places;
- T is a finite set of transitions, $P \cap T = \emptyset$; and
- $F \subseteq (P \times T) \cup (T \times P)$ is a set of directed arcs. □

Given a Petri net $N = (P, T, F)$ and a place $p \in P$, we use notation $\bullet p$ to represent a set of transitions taking place immediately before a place p, i.e., $\bullet p = \{t | t \in T \wedge (t, p) \in F\}$, we also call $\bullet p$ the pre-set of p. Notation $p\bullet$ is used to represent a set of transitions immediately after place p, i.e., $p\bullet = \{t | t \in T \wedge (p, t) \in F\}$, $p\bullet$ is also called the post-set of p.

The state of a Petri net $N = (P, T, F)$ is represented by its markings which is a distribution of tokens on places P. A marking of N is defined by a mapping function $m : P \to \mathbb{N}$, where $\mathbb{N}$ is the set of natural numbers. A place p is *marked* by a marking m if $m(p) > 0$.

The execution semantics of a Petri net $N = (P, T, F)$ are defined by transition firings which specify the enabling conditions and the marking transformations of the Petri net. A transition $t \in T$ is enabled by a marking m, if m marks all places in its pre-set $\bullet t$, i.e., $\forall p \in \bullet t : m(p) > 0$. In a Petri net $N = (P, T, F)$, with transition $t \in T$, and marking m, $(N, m) \xrightarrow{t}$ denotes that transition t is enabled at the marking m under the Petri net N. The firing of an enabled transition t transforms the marking m to m' as below:

$$m'(p) = \begin{cases} m(p) - 1 & \text{if} \quad p \in \bullet t \wedge p \notin t\bullet \\ m(p) + 1 & \text{if} \quad p \notin \bullet t \wedge p \in t\bullet \\ m(p) & \text{otherwise} \end{cases} \tag{1}$$

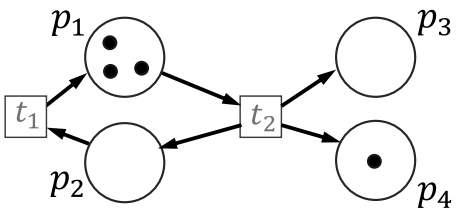

Fig. 2. A Petri net example.

Given a Petri net $N = (P, T, F)$, a transition $t \in T$, and two markings m, m', according to formula (1), $(N, m) \xrightarrow{t} (N, m')$ denotes that the enabled transition t results in marking m'. We use the similar way to represent the firing sequence of the transitions $t_1, \ldots, t_n$, i.e., $(N, m^{start}) \xrightarrow{t_1} \xrightarrow{t_2} \ldots \xrightarrow{t_n} (N, m^{end})$, where m^{start} is the initial marking, m^{end} is the new marking derived by firing the sequence of $t_1, \ldots, t_n$.

To explain on the definitions listed above, we use the Petri net shown in Fig. 2 to provide a more concrete example. In the graphical representation, *places*, *transitions*, *arcs*, and *tokens* are represented by circles, squares, arrows, and black dots respectively. The Petri net N shown in Fig. 2 is defined as (P, T, F), where $P = \{p_1, p_2, p_3, p_4\}$, $T = \{t_1, t_2\}$, and $F = \{(t_1, p_1), (p_1, t_2), (t_2, p_2), (t_2, p_3), (t_2, p_4), (p_2, t_1)\}$. The current marking is $m = \{(p_1, 3), (p_2, 0), (p_3, 0), (p_4, 1)\}$. Hence, the transition t_2 is enabled by the marking m denoted by $(N, m) \xrightarrow{t_2}$. According to formula (1), the firing of the transition t_2 transforms the marking m to the new marking m', i.e., $(N, m) \xrightarrow{t_2} (N, m')$, where $m' = \{(p_1, 2), (p_2, 1), (p_3, 1), (p_4, 2)\}$. Note that, for a marking m, if a place does not contain any token, i.e., $(p, 0)$, we omit it for simplicity. For instance, we can simplify $m = \{(p_1, 3), (p_2, 0), (p_3, 0), (p_4, 1)\}$ to $m = \{(p_1, 3), (p_4, 1)\}$.

Note that the process models that are mined by most of the existing algorithms [3,14,15] are subclasses of the Petri net . One such subclass known as a *workflow net* expands the Petri net by introducing three further constraints: (1) there is one and only one *input place* where a process starts, (2) there is one and only one *output place* where the process ends, and (3) all elements are on a path from the input place to the output place. The formal definition is given below.

Definition 5 (*Workflow Net [17]*)**.** A net $N = (P, T, F)$ is a workflow net, if it is a Petri net and satisfies the following constraints:

- There is one and only one input place i, i.e.
  1. $\exists i \in P$, s.t. $\forall t \in T, (t, i) \notin F$,
  2. $\exists i_1, i_2 \in P$, $(\forall t \in T, (t, i_1) \notin F \wedge (t, i_2) \notin F) \to i_1 = i_2$.
- There is one and only one output place o, i.e.
  1. $\exists o \in P$, s.t. $\forall t \in T, (o, t) \notin F$,
  2. $\exists o_1, o_2 \in P$, $(\forall t \in T, (o_1, t) \notin F \wedge (o_2, t) \notin F) \to o_1 = o_2$
- For a pseudo transition $\xi \notin T$, the net $(P, T \cup \{\xi\}, F \cup \{(o, \xi), (\xi, i)\})$ is a strongly connected net. □

It is worth pointing out that the three constraints neither change the syntax or the execution semantics of a Petri net . Since our work is based on the workflow nets that are derived from existing process mining algorithms, the process model refers to the workflow net in the later sections of this paper.

3. Discovering different scenarios in wastewater treatment processes

In this section, we develop an approach to discover the process scenarios from a given event log in wastewater treatment domain. In particular, we firstly obtain time dependent sets from a process model produced by applying an existing process discovery algorithm on the

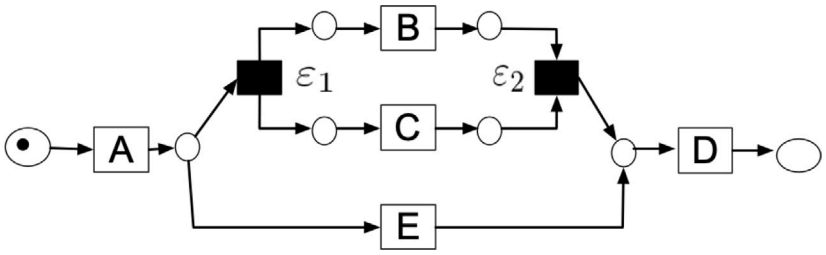

Fig. 3. A process model corresponding to the event log.

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. As mentioned in introduction, for a given event log, often times, we do not know aprior how many different scenarios are embedded in the log. Hence, for the third step, we apply a density-based approach to partition the event log into a set of sub-logs based on the aggregated vectors so that existing process discovery algorithms can be applied on the sub-logs to obtain different process scenarios in terms of process models.

3.1. Obtain time dependent sets from a process model

In the following, we first use an example to illustrate the steps to extract the time dependent set for each transition in a process model derived from a given event log file. We then give an algorithm that automates the steps.

Consider the event log shown in Table 1. We apply an existing process mining algorithm, for instance, the inductive miner algorithm developed by [18], on the event log. The obtained workflow net is depicted in Fig. 3. As shown in Fig. 3, the workflow net starts at task A and finish after task D. In the workflow net constructed by the inductive miner algorithm, there are invisible transitions represented in black rectangles, such as ε_1 and ε_2, in Fig. 3. Invisible transitions are only for routing purposes. They are produced by process mining algorithms, not represent recorded activities in event logs.

Before we show how to extract a time dependent set of a transition, we first give its formal definition.

Definition 6 (*Time Dependent Set*)**.** Given a workflow net $N = (P, T, F)$, where $T = A \cup \Gamma$, $A \cap \Gamma = \emptyset$. A is a set of non-invisible transitions, and Γ is a set of invisible transitions. The time dependent relation set $\Theta(t)$ of a transition $t \in A$ is defined as follows:

$$\Theta(t) = \{a \in A | (a \bullet \cap \bullet t \neq \emptyset) \vee (\exists \varepsilon_1, \ldots, \varepsilon_i \in \Gamma : a \bullet \cap \bullet \varepsilon_1 \neq \emptyset \wedge \varepsilon_1 \bullet \cap \bullet \varepsilon_2 \neq \emptyset \wedge \cdots \wedge \varepsilon_{i-1} \bullet \cap \bullet \varepsilon_i \neq \emptyset \wedge \varepsilon_i \bullet \cap \bullet t \neq \emptyset)\}$$

Example 1. Consider the event log given in Table 1 and a corresponding process model shown in Fig. 3, 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. 3. 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.

Algorithm 1 gives the procedure of obtaining the time dependent set for each activity in a process model. The time complexity of algorithm 1 is $O(N^3)$, where N is the number of transitions in the workflow net.

Algorithm 1 Extracting Time Dependent Set of a Transition

Input: a workflow net $N = (P, T, F)$, a transition x
Output: The time dependent set Θ of transition x

```
Define three sets Θ(x) = { } ; A = { } ; Γ = { }
for each transitions t ∈ T do
  if t is non-invisible transition then
    A = A ∪ {t}
  else
    Γ = Γ ∪ {t}
  end if
end for
for each transition t0 ∈ T do
  if •x ∩ t0• ≠ ∅ then
    Θ(x) = Θ(x) ∪ {t0}
  end if
end for
repeat
  Define break condition: condition = False
  for each transition t1 ∈ Θ(x) do
    if t1 ∈ Γ then
      condition = True
      Θ(x) = Θ(x) − {t1}
      for each transition t2 ∈ T do
        if •t1 ∩ t2• ≠ ∅ then
          Θ(x) = Θ(x) ∪ {t2}
        end if
      end for
    end if
  end for
until condition
return Θx
```

3.2. Construct aggregated vectors using time dependent set

For the second step, we 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 [10] to map each event trace σ in a given event log into an *activity vector* ($\vec{\mathcal{A}}_\sigma$), which is defined below.

Definition 7 (*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 $\vec{\mathcal{A}}_\sigma = (\mathcal{N}_1(\alpha_1), \ldots, \mathcal{N}_n(\alpha_i))$, 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 1, 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 piece of 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.

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

Definition 8 (*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 9 (*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 duration between two activities of a given time dependent pair $(\alpha, \beta) \in \mathcal{O}$. For a given trace $\sigma = [e_1, \dots, e_m]$, 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** $1(\sigma_1)$ in Table 1 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 2 gives the detailed steps of obtaining the timing vector from an event trace.

Algorithm 2 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
```

Before we aggregate the activity and timing vectors to obtain a vector that contains both activity information and timing information in an event trace, we normalize the activity and timing vectors. Normalization of the activity and timing vector is used to prevent features with large numerical values from dominating in distance-based objective functions which is used in trace clustering. For a given event trace σ, the corresponding activity vector and timing vector are $\vec{\mathcal{A}} = (a_1, \dots, a_m)$ and $\vec{\mathcal{T}} = (t_1, \dots, t_n)$, respectively, we apply $\frac{\vec{\mathcal{A}}}{\|\vec{\mathcal{A}}\|}$ and $\frac{\vec{\mathcal{T}}}{\|\vec{\mathcal{T}}\|}$ to normalize the vectors, where $\|\vec{\mathcal{A}}\| = \sqrt{a_1^2 + \cdots + a_n^2}$ and $\|\vec{\mathcal{T}}\| = \sqrt{t_1^2 + \cdots + t_n^2}$. Finally, we aggregate the normalized activity and normalized timing vectors.

Table 2
Aggregated vectors transferred from the event log presented in Table 1.

	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)
$\vec{\mathcal{V}}_{\sigma_1}$	0.500	0.500	0.500	0.500	0.000	0.419	0.423	0.569	0.566	0.000	0.000
$\vec{\mathcal{V}}_{\sigma_2}$	0.500	0.500	0.500	0.500	0.000	0.759	0.458	0.004	0.439	0.000	0.000
$\vec{\mathcal{V}}_{\sigma_3}$	0.500	0.500	0.500	0.500	0.000	0.144	0.742	0.652	0.053	0.000	0.000
$\vec{\mathcal{V}}_{\sigma_4}$	0.500	0.500	0.500	0.500	0.000	0.624	0.322	0.305	0.642	0.000	0.000
$\vec{\mathcal{V}}_{\sigma_5}$	0.577	0.000	0.000	0.577	0.577	0.000	0.000	0.000	0.000	0.151	0.988

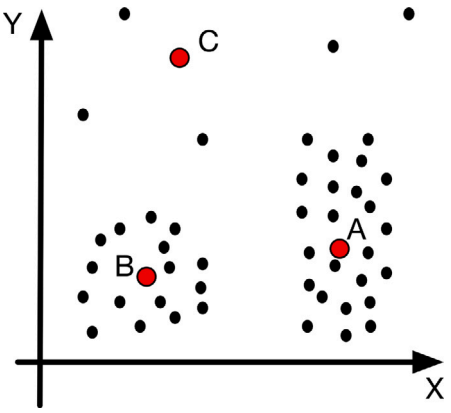

Fig. 4. Illustration of the strategy about the density-based algorithm.

Definition 10 (*Aggregated Vector* $(\vec{\mathcal{V}})$)**.** Given an event trace σ, assume the corresponding activity and timing vectors are $\vec{\mathcal{A}}_\sigma = (a_1, \dots, a_i, \dots, a_m)$ and $\vec{\mathcal{T}}_\sigma = (t_1, \dots, t_j, \dots, t_n)$, respectively, the aggregated vector $\vec{\mathcal{V}}$ corresponding to an event trace σ is $\vec{\mathcal{V}}_\sigma = (\frac{a_1}{\|\vec{\mathcal{A}}\|}, \dots, \frac{a_i}{\|\vec{\mathcal{A}}\|}, \dots, \frac{a_m}{\|\vec{\mathcal{A}}\|}, \frac{t_1}{\|\vec{\mathcal{T}}\|}, \dots, \frac{t_j}{\|\vec{\mathcal{T}}\|}, \dots, \frac{t_n}{\|\vec{\mathcal{T}}\|})$, where $a_i = \vec{\mathcal{A}}_\sigma[i]$, $t_j = \vec{\mathcal{T}}_\sigma[j]$, $\|\vec{\mathcal{A}}\| = \sqrt{a_1^2 + \cdots + a_n^2}$, and $\|\vec{\mathcal{T}}\| = \sqrt{t_1^2 + \cdots + t_n^2}$.

Consider **Trace** $1(\sigma_1)$ in Table 1, the aggregated vector corresponding to σ_1 is $\vec{\mathcal{V}}_{\sigma_1} = (0.5, 0.5, 0.5, 0.5, 0, 0.419, 0.423, 0.569, 0.566, 0, 0)$. Table 2 gives the vector value for the other four traces. Note that, we can also use weighted concatenation to aggregate the normalized activity and timing vectors. The weighted values provide a way to emphasize which perspective should have more impact on the outcome of discovery process scenarios.

3.3. Discover process scenarios with density based clustering algorithm

Given an event log L, we first construct the set of aggregated vectors $\{\vec{\mathcal{V}}_1, \dots, \vec{\mathcal{V}}_n\}$, we then apply the density-based algorithm to partition the event log into a set of sub logs. Finally, existing process discovery algorithms can be applied on the sub logs to obtain different process scenarios in terms of process models.

We use Fig. 4 to illustrate the basic idea used in the density-based algorithm to partition the event log into a set of clusters. For simplicity, assume the cardinality of event trace's aggregated vector $|\vec{\mathcal{V}}| = 2$, therefore, each aggregated vector (an event trace) can be mapped to a point in a two-dimensional domain shown in Fig. 4.

Consider the sample sets of aggregated vectors depicted in Fig. 4, we can easily and clearly detect two different process scenarios of aggregated vectors and noise not belonging to any of those scenarios. For instance, aggregated vector A and B belong to two different process scenarios. However, the aggregated vector C is noise. The main reason why we recognize the clusters is that within each cluster, we have a typical density of points which is considerably higher than outside of the cluster. Furthermore, the density within the areas of noise is lower than the density in any of the clusters. The key idea is that

for each point of a cluster the neighborhood of a given radius has to contain at least a minimum number of points, which the density in the neighborhood has to exceed a threshold. Hence, we would have to define the two appropriate parameters which are the following. The distance that will be used to locate the points in the neighborhood of any point and the minimum number of points clustered together for a region to be considered dense. Then, we can retrieve all points that are reachable from the given point using the these two parameters. There is no easy way to get the knowledge related to the parameters in advance, the existing technique in the literature to determine the parameters of the cluster is heuristic [19–21]. Sometimes, the two parameters can be set by domain experts [9].

Let $d(\vec{\mathcal{V}}_i, \vec{\mathcal{V}}_j)$ denote *Euclidean* distances between aggregated vectors $\vec{\mathcal{V}}_i$, $\vec{\mathcal{V}}_j$. Let Eps denote the distance measure that will be used to locate the points in the neighborhood of any point. Let $MinPts$ denote the minimum number of points clustered together for a region to be considered dense.

Algorithm 3 shows the procedure to partition the event log into a set of clusters using the density-based algorithm from aggregated vectors. The time complexity of the algorithm is $O(n \times log(n))$, where n is the number of event traces in the event log L. We use Example 2 to illustrate Algorithm 3.

Algorithm 3 Event log partitioning

Input: A set of aggregated vectors $\Pi = \{\vec{\mathcal{V}}_1, \cdots, \vec{\mathcal{V}}_{|L|}\}$ and a set of *Euclidean* distances $\mathcal{M} = \{d(\vec{\mathcal{V}}_i, \vec{\mathcal{V}}_j) | \vec{\mathcal{V}}_i, \vec{\mathcal{V}}_j \in \Pi\}$; Eps and $MinPts$ be the two parameters.
Output: a set of the clusters related to aggregated vectors

1: $k \leftarrow 0$
2: **for each** $\vec{\mathcal{V}}_i \in \Pi$ **do**
3: **if** $\vec{\mathcal{V}}_i$ is not labeled **then**
4: **continue**
5: **end if**
6: $NeighborsList \leftarrow$ Empty List
7: **for each** $\vec{\mathcal{V}}_j \in \Pi$ **do**
8: **if** $d(\vec{\mathcal{V}}_i, \vec{\mathcal{V}}_j) \leq Eps$ **then**
9: $NeighborsList \leftarrow NeighborsList \cup \{\vec{\mathcal{V}}_j\}$
10: **end if**
11: **end for**
12: **if** $|NeighborsList| < MinPts$ **then**
13: $\vec{\mathcal{V}}_i$ is labeled as noise
14: **continue**
15: **end if**
16: $k \leftarrow k + 1$ and $\vec{\mathcal{V}}_i$ is labeled as k
17: $SearchList \leftarrow NeighborsList \backslash \{\vec{\mathcal{V}}_i\}$
18: **for each** $\vec{\mathcal{V}}_m \in SearchList$ **do**
19: **if** $\vec{\mathcal{V}}_m$ is labeled as noise **then**
20: $\vec{\mathcal{V}}_m$ is labeled as k
21: **end if**
22: $\vec{\mathcal{V}}_m$ is labeled as k
23: $NeighborsList \leftarrow$ Empty List
24: **for each** $\vec{\mathcal{V}}_n \in \Pi$ **do**
25: **if** $d(\vec{\mathcal{V}}_m, \vec{\mathcal{V}}_n) \leq Eps$ **then**
26: $NeighborsList \leftarrow NeighborsList \cup \{\vec{\mathcal{V}}_n\}$
27: **end if**
28: **end for**
29: **if** $|NeighborsList| \geq MinPts$ **then**
30: $SearchList \leftarrow SearchList \cup NeighborsList$
31: **end if**
32: **end for**
33: **end for**

Table 3
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 [23]	819	2351	2.871
BPIC2020 [24]	10 500	56,437	5.374
Hospital billing [25]	100 000	451 359	4.514
Review process [26]	10 000	236 360	23.636
Road traffic [27]	150 370	561 470	3.734

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 2, the set of *Euclidean* distances of two aggregated vectors in L is $\mathcal{M} = \{(\vec{\mathcal{V}}_1, \vec{\mathcal{V}}_2) = 0.203, (\vec{\mathcal{V}}_1, \vec{\mathcal{V}}_3) = 0.202, (\vec{\mathcal{V}}_1, \vec{\mathcal{V}}_4) = 0.108, (\vec{\mathcal{V}}_1, \vec{\mathcal{V}}_5) = 0.508, (\vec{\mathcal{V}}_2, \vec{\mathcal{V}}_3) = 0.306, (\vec{\mathcal{V}}_2, \vec{\mathcal{V}}_4) = 0.124, (\vec{\mathcal{V}}_2, \vec{\mathcal{V}}_5) = 0.507, (\vec{\mathcal{V}}_3, \vec{\mathcal{V}}_4) = 0.281, (\vec{\mathcal{V}}_3, \vec{\mathcal{V}}_5) = 0.508, (\vec{\mathcal{V}}_4, \vec{\mathcal{V}}_5) = 0.508\}$, and the two parameters are $Eps = 0.3$ and $MinPts = 1$.

According to Lines 2–5, we select an aggregated vector $\vec{\mathcal{V}}_1$, and check the label of aggregated vector $\vec{\mathcal{V}}_1$. We apply Line 6–11 of algorithm for every aggregated vector in the set Π to find the neighborhood of aggregated vector $\vec{\mathcal{V}}_1$. 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 0.203 and less than Eps, we add $\vec{\mathcal{V}}_2$ into list $NeighborsList$. After applying the step to the other three aggregated vectors $\vec{\mathcal{V}}_3, \vec{\mathcal{V}}_4, \vec{\mathcal{V}}_5$, the $NeighborsList$ becomes $\vec{\mathcal{V}}_2, \vec{\mathcal{V}}_3, \vec{\mathcal{V}}_4$. Then we check whether the list $NeighborsList$ contains at least a $MinPts$ of points on Line 12–15. Based on Line 16 and 17, aggregated vector $\vec{\mathcal{V}}_1$ is labeled as 1 and then removed from list $NeighborsList$. We apply Line 18–32 of algorithm to find the neighborhood of every aggregated vector in the list $NeighborsList$, i.e. $\vec{\mathcal{V}}_2, \vec{\mathcal{V}}_3, \vec{\mathcal{V}}_4$. We repeat the steps until all aggregated vectors are labeled, and partition the event log into two subsets, i.e., scenarios. The first subset of the event log contain $\vec{\mathcal{V}}_1, \vec{\mathcal{V}}_2, \vec{\mathcal{V}}_3, \vec{\mathcal{V}}_4$, and the second subset contain $\vec{\mathcal{V}}_5$

Finally, the existing process discovery algorithms can be applied on the subset of logs to obtain different process scenarios in terms of process models.

The proposed techniques of process scenarios discoveries in wastewater treatment process is implemented on an existing open-source process mining framework *pm4py* [22]. The implementation of the technique is available at https://github.com/stevenzzy9/Wastewater-Treatment-Process-Discovery.

4. Experimental study

In this section, we firstly apply the technique developed in earlier sections to the artificial logs to investigate the effectiveness and validity. Then we experimentally evaluate the proposed approach on five real-life event logs [23–27]. The characteristics of these event logs are summarized in Table 3. The artificial logs are generated from the Processes and Logs Generator (**Plg**) [28], which is an open-source tool that is used to generate artificial event logs based on the user-defined process model. The objectives of our evaluations consist of two components. First, investigate the effectiveness in process scenarios discoveries in wastewater treatment process. To do so, we apply the proposed approach on the artificial logs, which is generated from the **Plg** based real wastewater treatment process provided by a domain expert and examine the effectiveness of the approach. Second, we evaluate if having timing information can improve the quality of process scenario models discovered from event logs. In order to achieve the objective, we choose an existing heuristic mining algorithm [12], and apply it on the same event logs but with and without time information. Before the evaluation, we first define our evaluation criteria.

4.1. 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 [29]. The higher fitness value and precision value, the better quality of the process model. We use the pm4py [22] 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 [30], 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 [31] 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.

4.2. Effectiveness validation

This set of experiments is to examine the effectiveness of the proposed approach for process scenario discovery in wastewater treatment domain. More specifically, based on the process model provided by the domain expert, we first use the **Plg** to generate five artificial event logs by injecting an incremental amount of noise ranging from 1% to 20% to produce an event log that contains one thousand event traces. Then, for each artificial event log, we apply the proposed approach in Section 3 to partition the event log into clusters, and discover a process model from each subset of the event log using the heuristic mining algorithm presented in [12].

Fig. 5 shows the wastewater treatment process model in the form of the Business Process Modeling Notation (**BPMN**) [32]. All wastewater treatment processes begin with influent wastewater arriving at a treatment facility from sewage lines. The wastewater is sent through coarse and fine screens in order to remove both large and small objects, debris, etc (respectively) from the water. After these first mandatory steps are completed, the wastewater will go through different sub-processes depending on if the water can be reused or if it will be disposed of at a landfill.

Figs. 6, 7, and 8 show the results produced by the proposed approach. The first scenario within the wastewater treatment process shown in Fig. 6 is used to remove solid debris in the water, that range anywhere from 0.01 in to 6 in. This process begins at the "Coarse Screens" activity where the screens have openings of 0.25 in to 6 in. These screens are used to remove the larger solids, such as rags, sticks, rocks, etc. After the large debris is removed from the wastewater, it moves to the "Fine Screens" activity where smaller particles are moved from the water. Screens anywhere from 0.01 in to 0.25 in are used to suspend solids in the water and clarify the water to acceptable levels. The final processing step is the "Compacted Screenings" where after the screens used previously are washed, they are then compacted to reduce the amount of water and increase the concentration of solids. The first scenario ends with the activity "Disposal to Landfill" where waste is disposed of.

The second scenario within the wastewater treatment process shown in Fig. 7 is used to treat sludge by converting to a solid form. The same process of water going through the coarse/fine screens occurs the same as the first scenario. Then, the water goes through the "Primary Clarifiers" activity where solids are separated from the wastewater by gravity in a clarification tank. After this activity, "Blending" routes some of the water flow away from the treatment process so that it can be blended with fully treated water later on before it is disposed of. The "Anaerobic Digester" step continues the treatment process with degrading organic matter by removing oxygen. The process can either end at this point, or continue with the "Storage Tank" activity where water can be stored before it is disposed of at a landfill.

The third scenario within the wastewater treatment process shown in Fig. 8 is used to describe the liquid flow through the process. The third scenario also starts with coarse/fine screen activities and then goes to the "Primary Clarifiers" activity, similar to the second scenario, to allow for solids to separate from the liquids. The liquid then moves to the "Bio-reactor Tanks" where the liquid is treated by microfiltration membrane bioreactors (MBR's) that combine a suspended growth biological reactor along with microfiltration for secondary treatment of the water. Then, depending on various chemical levels, clarity levels, etc. the water will simultaneously travel to various other activities before the end of the process. The first route is to the "Final Clarifier" activity where an activated sludge process is used to settle out microorganisms. Then the "Chlorine Contact Basin" is also used to disinfect the water and remove microorganisms, bacteria, viruses, etc. from the water. Water is then moved to "Ponds" where organic content and pathogens are reduced/removed from the wastewater. Then the "De-chlorination Pond" is used to reduce the levels of chlorine in the wastewater. Before the process ends, the water is moved to either the "Effluent Pumping Station" for transporting the water, or the Cascade Aerator to oxidize iron and reduce dissolved gases. The second route within the third scenario starts off with one of three activities. The first, "Gravity Thickening" is used to condense biosolids in order to separate the solids from the solid free supernatant. The second activity, "Dissolved-Air Flotation Thickening", thickens sludge to bring solids to the surface and therefore separates the solids and liquids. The final option is "Wastewater Centrifuge De-watering Thickening" where a high speed rotation process separates water from the sewage. After one of these processes is used to treat the wastewater, it is then sent to the "Blend Tank" and the "Anaerobic Digester" for further treatment before the process ends.

According to the wastewater treatment workflow from conventional water reclamation plants, the wastewater treatment is a process in which the solids in wastewater are partially removed and partially changed by decomposition from highly complex, putrescible, organic solids to mineral or relatively stable organic solids. Primary and secondary treatment removes the majority of BOD and suspended solids found in wastewaters. The proposed approach utilize the duration of each treatment activities and the frequence of event trace sequences related to treatment processes for different waste forms to discover the process scenarios in wastewater treatment. Based on the domain expert's verification, the evaluation results show that the proposed approach is able to discover the process scenarios from event logs in wastewater treatment domain.

4.3. Performance evaluation

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 3.2 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 [10] to obtain the activity vectors that contain only the activity information. For both aggregated vectors and activity

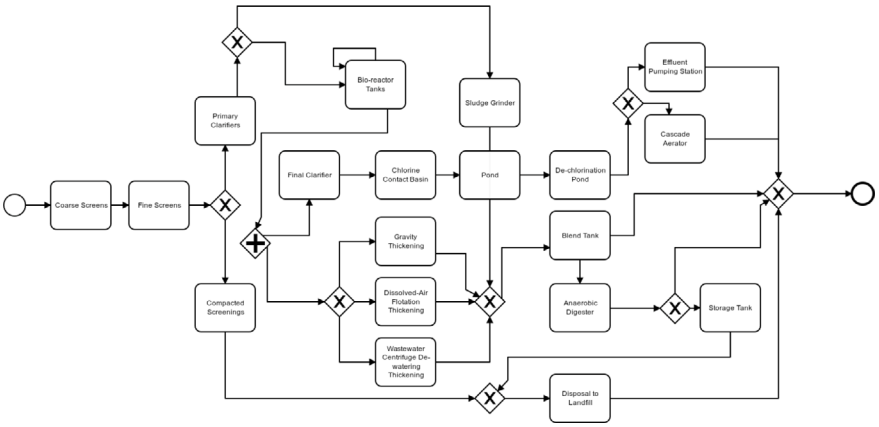

Fig. 5. Wastewater treatment process model.

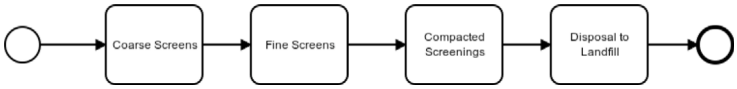

Fig. 6. Scenario one: Removing solid debris.

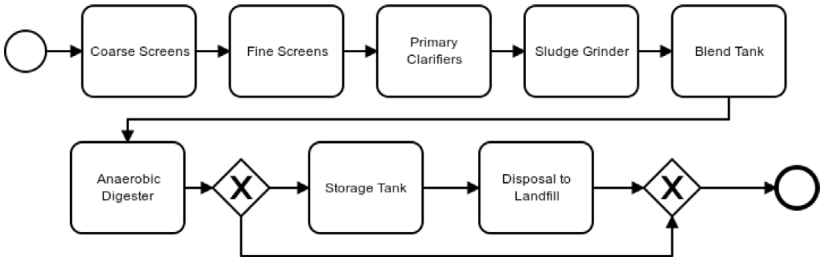

Fig. 7. Scenario two: Treating sludge.

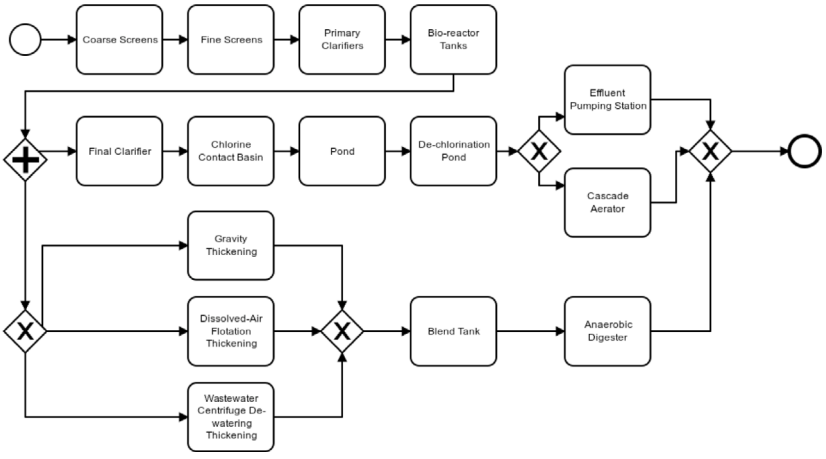

Fig. 8. Scenario three: Treating liquid.

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 [4].

We use the pm4py tool [22] 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 Tables 4, 5, and 6, 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 7. 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

Table 4
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 5
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 6
$F1$ 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 7
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%

average fitness resulted from the approach in [10] is closer to 1 than the weighted average precision, which indicates the room for fitness improvement is relatively small.

4.4. Comparison

This set of experiments is to compare our proposed approach with the four commonly used process scenarios discovery approaches such as the *k-means* approach (**KM**) [10], the *Model-based* approach (**MB**) [30], the *Distanced-based* approach (**DB**) [9] and the *Conserved Patterns* approach (**CP**) [11] using real-life event logs.

For each real-life event logs, we use the different approaches to generate subsets of event logs where event traces in the same subset are most likely under the same scenarios. For each subset of logs produced by the different approaches, we apply the inductive mining algorithm to discover a process model, and calculate the weighted average fitness, the weighted average precision, and the $F1$ score of the process models. The results are depicted in Table 8.

From the experiment results, the performance of the proposed approach is 3.31%, 7.93%, and 5.61% higher than the commonly approaches with respectively for *fitness*, *precision*, and *F1 score* on average. The proposed method outperforms four commonly used approaches with different real-life event logs.

5. Related work

Over the past decade, a range of effective process discovery algorithms have been proposed [33] in process mining field. Despite the demonstrated usefulness of process discovery algorithms, these algorithms face challenges in an environment where different scenarios exist [5–8]. When different scenarios are grouped into one process model, the complexity of the model becomes incomprehensible. The work in [34] is the first study which applies 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

Table 8
Weighted average fitness, weighted average precision, and F1 score.

	BPIC 2013			BPIC 2020			Review process			Toad traffic			Hospital billing		
	Weighted average fitness	Weighted average precision	F1 score	Weighted average fitness	Weighted average precision	F1 score	Weighted average fitness	Weighted average precision	F1 score	Weighted average fitness	Weighted average precision	F1 score	Weighted average fitness	Weighted average precision	F1 score
Proposed approach	0.985	0.877	0.928	0.998	0.887	0.939	0.951	0.646	0.769	0.974	0.719	0.827	0.936	0.793	0.859
KM approach	0.951	0.816	0.878	0.985	0.883	0.931	0.898	0.530	0.667	0.768	0.640	0.698	0.785	0.792	0.788
DB approach	0.952	0.721	0.821	0.986	0.876	0.928	0.915	0.431	0.586	0.823	0.606	0.698	0.973	0.774	0.862
CP approach	0.965	0.675	0.794	0.986	0.872	0.926	0.925	0.475	0.628	0.963	0.561	0.709	0.945	0.619	0.748
MB approach	0.972	0.651	0.780	0.989	0.889	0.936	0.939	0.492	0.646	0.969	0.561	0.711	0.931	0.663	0.774

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 [35]. Greco et al. [9] 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. [10] proposed an approach to construct a vector space model for traces in an event log. In [10], the author allows for different kinds of attributes, e.g., activity name, executor, to determine the vector associated with each event trace, and then provide four clustering techniques for trace clustering. In [6,11], 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. [6] 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. [11] 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 [36]. In [36], 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.

In the wastewater treatment domain, the demands of wastewater treatment facilities have increased and facilities require methods to improve their treatment processes. For example, regulatory demands from government bodies [37], such as the EPA, require wastewater to meet increasingly strict quality standards in order to be safe. One approach to solve this issue is by creating wastewater treatment process models in order to analyze and improve these facilities' processes. In the domain of wastewater treatment, there are a variety of tools and methods generally used in the modeling of treatment processes. One popular tool, GPS-X, is often used in modeling [38,39] because it is specifically designed for wastewater modeling and simulation. While this does provide the user some level of convenience, these software tools lack due to the need of frequent human input [38], levels of uncertainty [37], among other factors. Our approach differs in the aspect of how the data is processed and used to create models. By using timing information to discover process scenarios, we lower the levels of uncertainty in that the process models should more accurately reflect the actual, real world scenarios occurring in the treatment process.

6. Conclusion

In this paper, we present an approach that uses timing information to assist in discovering process scenarios from event logs in wastewater treatment processes without a prior knowledge about the number of scenarios k. The algorithm is implemented based on the open-source process mining framework *pm4py*. A real wastewater treatment process provided by a domain expert is used as a case study to investigate the effectiveness and validity of the approach. The evaluation results show that the algorithm is able to discover the process scenarios from event logs in wastewater treatment domain. The experiments on real-life event logs show that the process scenario models obtained with the additional timing information have higher fitness and precision scores than the models obtained without the timing information. These results lead to the conclusions that timing information can improve process scenario discovery performance, and the proposed approach can discover different process scenarios more effectively in wastewater treatment process. From the experiment results, we also notice that the scenario discovery process is 'mechanical' in the sense that it does not have domain experts involved or utilize domain experts' knowledge in the process. In future works, we will study how domain knowledge may be represented and utilized in scenario discoveries.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

We thank Dr. Junjie Zhu (Princeton University) for providing the real wastewater treatment process model. This work is supported in part by NSF 1952247, NSF 1952225, and NSF 1929469.

References

[1] Wil Van der Aalst, Ton Weijters, Laura Maruster, Workflow mining: Discovering process models from event logs, IEEE Trans. Knowl. Data Eng. 16 (9) (2004) 1128–1142.
[2] Marc Solé, Josep Carmona, Process mining from a basis of state regions, in: International Conference on Applications and Theory of Petri Nets, Springer, 2010, pp. 226–245.
[3] Robin Bergenthum, Jörg Desel, Robert Lorenz, Sebastian Mauser, Process mining based on regions of languages, in: International Conference on Business Process Management, Springer, 2007, pp. 375–383.
[4] A.J.M.M. Weijters, Wil M.P. van Der Aalst, A.K. Alves De Medeiros, Process Mining with the Heuristics Miner-Algorithm, Vol, 166, Tech. Rep. WP, Technische Universiteit, Eindhoven, 2006, pp. 1–34.
[5] Stijn Goedertier, Jochen De Weerdt, David Martens, Jan Vanthienen, Bart Baesens, Process discovery in event logs: An application in the telecom industry, Appl. Soft Comput. 11 (2) (2011) 1697–1710.
[6] R.P. Jagadeesh Chandra Bose, Wil M.P. Van der Aalst, Context aware trace clustering: Towards improving process mining results, in: Proceedings of the 2009 SIAM International Conference on Data Mining, SIAM, 2009, pp. 401–412.
[7] Gabriel M. Veiga, Diogo R. Ferreira, Understanding spaghetti models with sequence clustering for ProM, in: International Conference on Business Process Management, Springer, 2009, pp. 92–103.
[8] Christian W. Günther, Process Mining in Flexible Environments, Technische Universiteit, Eindhoven, 2009.
[9] .
[10] Minseok Song, Christian W. Günther, Wil M.P. Van der Aalst, Trace clustering in process mining, in: International Conference on Business Process Management, Springer, 2008, pp. 109–120.
[11] R.P. Jagadeesh Chandra Bose, Wil M.P. van der Aalst, Trace clustering based on conserved patterns: Towards achieving better process models, in: International Conference on Business Process Management, Springer, 2009, pp. 170–181.
[12] Anton J.M.M. Weijters, Wil M.P. Van der Aalst, Rediscovering workflow models from event-based data using little thumb, Integr. Comput.-Aided Eng. 10 (2) (2003) 151–162.
[13] Wil Van Der Aalst, Process Mining: Discovery, Conformance and Enhancement of Business Processes, Vol. 2, Springer, 2011.
[14] A.J.M.M. Weijters, Wil M.P. van der Aalst, Process mining: discovering workflow models from event-based data, in: Belgium-Netherlands Conf. on Artificial Intelligence, Citeseer, 2001.
[15] Christian W. Günther, Wil M.P. Van Der Aalst, Fuzzy mining–adaptive process simplification based on multi-perspective metrics, in: International Conference on Business Process Management, Springer, 2007, pp. 328–343.

[16] Jörg Desel, Wolfgang Reisig, Place/transition Petri Nets, in: Lectures on Petri Nets I: Basic Models: Advances in Petri Nets, Springer Berlin Heidelberg, Berlin, Heidelberg, 1998, pp. 122–173.
[17] Wil M.P. Van der Aalst, A.J.M.M. Weijters, Laura Maruster, Workflow Mining: Which Processes can be Rediscovered, Technical report, BETA Working Paper Series, WP 74, Eindhoven University of Technology, Eindhoven, 2002.
[18] Alejandro Bogarín Vega, Rebeca Cerezo Menéndez, Cristobal Romero, Discovering learning processes using inductive miner: A case study with learning management systems (LMSs), Psicothema (2018).
[19] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise, in: Kdd, Vol. 96, 1996, pp. 226–231.
[20] Soumaya Louhichi, Mariem Gzara, Hanène Ben Abdallah, A density based algorithm for discovering clusters with varied density, in: 2014 World Congress on Computer Applications and Information Systems (WCCAIS), IEEE, 2014, pp. 1–6.
[21] Vivek S. Ware, H.N. Bharathi, Study of density based algorithms, Int. J. Comput. Appl. 69 (26) (2013).
[22] State-of-the-art-process mining in Python. https://pm4py.fit.fraunhofer.de/.
[23] BPI challenge 2013. https://data.4tu.nl/articles/dataset/BPI_Challenge_2013_incidents/12693914/1.
[24] BPI challenge 2020. https://data.4tu.nl/collections/BPI_Challenge_2020/5065541.
[25] Hospital billing - Event log. https://data.4tu.nl/articles/dataset/Hospital_Billing_-_Event_Log/12705113.
[26] Synthetic event logs - review example. https://data.4tu.nl/articles/dataset/Synthetic_event_logs_-_review_example_large_xes_gz/12716609.
[27] Road traffic fine management process. https://data.4tu.nl/articles/dataset/Road_Traffic_Fine_Management_Process/12683249.
[28] Andrea Burattin, PLG2: Multiperspective process randomization with online and offline simulations, in: BPM (Demos), 2016, pp. 1–6.
[29] Raffaele Conforti, Marcello La Rosa, Arthur H.M. ter Hofstede, Filtering out infrequent behavior from business process event logs, IEEE Trans. Knowl. Data Eng. 29 (2) (2016) 300–314.
[30] Jochen De Weerdt, Seppe Vanden Broucke, Jan Vanthienen, Bart Baesens, Active trace clustering for improved process discovery, IEEE Trans. Knowl. Data Eng. 25 (12) (2013) 2708–2720.
[31] Wil Van Der Aalst, Data science in action, in: Process Mining, Springer, 2016, pp. 3–23.
[32] Stephen A. White, Introduction to BPMN, Vol. 2, Ibm Cooperation, 2004.
[33] Adriano Augusto, Raffaele Conforti, Marlon Dumas, Marcello La Rosa, Fabrizio Maria Maggi, Andrea Marrella, Massimo Mecella, Allar Soo, Automated discovery of process models from event logs: Review and benchmark, IEEE Trans. Knowl. Data Eng. 31 (4) (2018) 686–705.
[34] Francesco Folino, Gianluigi Greco, Antonella Guzzo, Luigi Pontieri, Mining usage scenarios in business processes: Outlier-aware discovery and run-time prediction, Data Knowl. Eng. 70 (12) (2011) 1005–1029.
[35] Anil K. Jain, Richard C. Dubes, Algorithms for Clustering Data, Prentice-Hall, Inc., 1988.
[36] Annalisa Appice, Donato Malerba, A co-training strategy for multiple view clustering in process mining, IEEE Trans. Serv. Comput. 9 (6) (2015) 832–845.
[37] E. Belia, Y. Amerlinck, Lorenzo Benedetti, B. Johnson, Gürkan Sin, Peter A. Vanrolleghem, K.V. Gernaey, S. Gillot, M.B. Neumann, L. Rieger, et al., Wastewater treatment modelling: dealing with uncertainties, Water Sci. Technol. 60 (8) (2009) 1929–1941.
[38] H.M. Phillips, K.E. Sahlstedt, K. Frank, J. Bratby, W Brennan, S. Rogowski, D. Pier, W. Anderson, M. Mulas, J.B. Copp, et al., Wastewater treatment modelling in practice: a collaborative discussion of the state of the art, Water Sci. Technol. 59 (4) (2009) 695–704.
[39] Amra Serdarevic, Alma Dzubur, Wastewater process modeling, Coupled Syst. Mech. 5 (1) (2016) 21–39.