2021 18th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE).
Mexico City, Mexico. November 10-12, 2021

Discovering Structurally Simple Workflow Nets by Vector-Based Trace Clustering

César Barrón-Rubio, Ernesto López-Mellado
CINVESTAV, Unidad Guadalajara
Av. Del Bosque 1145. Col El Bajio 45019 Zapopan Jal. México
{cesar.barron, e.lopez}@cinvestav.mx

***Abstract*— Nowadays, models play a relevant role in the analysis and monitoring of processes carried out by organizations. Process discovery deals with the automated building of models from logs of event traces, which represent the executions of actual processes. However, when the logs come from highly flexible environments, most of the discovery algorithms obtain excessively complex models, which makes difficult their analysis. Trace clustering is a suitable approach to deal with this problem; it consists of dividing the logs into subsets, such that the executions within the same set are consistent with each other and derive a model for each subset. However, many clustering methods are based on the semantics of the executions, which is not the necessary feature for simplifying the model. In this paper, we propose a vector-based trace clustering approach in which the event traces are encoded as vectors, with the aim to gather the traces generated by the same process variant. The idea behind the method is that the different variants of a process can be distinguished by the corresponding sequential model substructures. Therefore, the proposed encoding method strives to partition the log into the subsets of traces from which those substructures can be revealed. An experimental evaluation with actual event logs shows that the method discovers simpler workflow nets than other available techniques.**

Keywords*— *Event-trace clustering, Process discovery, Workflow nets, Structural complexity reduction.

I. Introduction

Process Mining (PM) is a relatively recent research area, which deals with "discovering, monitoring, and improving real processes by extracting knowledge from the event logs available in the current system." [1]. An event log stores the behavior exhibited by the process as sequences or traces of event names. An event can save information about the identifier of the case to which it belongs, its order of execution, the name of the activity carried out, the start and end time, the resources used, etc. however, we only consider the identifier of the case, the order, and the name of the activity. In general, the event log is a multiset of traces.

The main branch of PM is Process discovery (PD); its purpose is to build, automatically, a model of the process by processing the event log. This modelling resource has shown a significant value primarily when using event logs obtained from well-structured environments [2]. However, most discovery methods yield structurally complex models when dealing with logs obtained from processes in highly flexible environments. This feature makes it difficult to analyze the models for extracting helpful information for reverse engineering purposes; for this reason, they are called spaghetti-like models [3].

C. Barrón-Rubio is financed by CONACYT (Mexico) through the Ph.D. grant No. 854470.

The problem of reducing the structural complexity of discovered Petri nets has been addressed through an trace classification approach, in which the event log λ is partitioned into sets of traces owning a common feature; then, a model is discovered for every set and the obtained models are simpler than the model obtained from λ. This strategy is known as *Trace clustering* [2].

Early trace clustering proposals include unsupervised learning algorithms [4], which use a metric to quantify the degree of similarity between two traces. Approaching trace clustering as an unsupervised learning problem has advantages and inconveniences. On the one hand, being the clustering problem a widely studied research topic, a plethora of algorithms and methods are available. On the other hand, these techniques are too data-centric, so the results are not always in line with the goals of process mining.

There are mainly two approaches that tackle this problem through unsupervised learning techniques.

The most direct approach is called *Trace Distance-Based Method*. It is supported by a metric between traces, which is usually based on the count of the necessary operations to transform one trace into another one. The best-known example is the Levenshtein distance, generalized in [5], although any distance between sequences can be used. An alternative approach is the *Vector Distance-Based Method*, in which the traces are encoded as vectors, and uses standard metrics. An earlier work following this approach can be found in [2], where the concept of trace profile is introduced; a trace is represented through its directly followed event relationships. Recently, in [6], [7], Deep Learning techniques are used to obtain meaningful representations of the traces.

The success of approaches based on a distance between traces depends to some extent on whether the metric used can detect the characteristics that reveal that two traces come from structurally similar processes. On the other hand, methods based on distances between vectors rely on the way the trace is encoded; vectors must encode the key features of a trace.

Trace encoding methods are mainly based on representation techniques used in *natural language processing*, known as *word-embeddings*. However, all these techniques are focused on semantics. Nevertheless, the complexity of a model is a structural characteristic, i.e., syntax. Thus, two traces can be semantically very similar and still come from processes with very different syntax. Therefore, a model obtained from a set of semantically similar traces can still be very complex.

This paper presents COBE (Conjoint Occurrence Based Encoding), a novel trace encoding method that strives to capture pertinent information about the underlying process structure. We hypothesize that the sequential substructures of the process variants allow us to differentiate the traces produced by each variant. The sequential substructures can be extracted from the event log using Conjoint Occurrence Classes (CO) [8]. COBE is based on the same trace handlers that COs, bringing the information of the COs to the vector representation. Therefore, traces in which a similar set of sequential substructures occur will be transformed into close vectors. This enables that clustering methods get "structurally similar" trace clusters.

The experimental evaluation was carried out using real event logs, and COBE showed a good performance in discrimination of process, comparable to the reached by the encodings based on the semantics of the traces, in accordance with adjusted rand and normalized mutual information scores. However, for the specific goal of reducing the complexity of the model, particularly the reduction of the complexity of the workflow models, COBE achieved a significantly better performance in accordance with complexity metrics specialized in evaluating this class of model.

II. Preliminaries

This section includes concepts and notation of Petri nets and basic notions on process discovery and trace clustering used in this paper.

A. Petri Nets

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

Given a Petri net $N = (P, T, F)$ and $x \in T \cup P$, we call *preset* of x to the set $\bullet\, x = \{y | (y, x) \in F\}$. Respectively, the *postset* of x is $x \bullet = \{y | (x, y) \in F\}$. In a marked Petri net (N, M), the transition $t \in T$ can be fired, iff $\bullet\, t \leq M$. The fired of t leads to a new marking $M' = (M \backslash \bullet\, t) \uplus t \,\bullet$. The set of all reachable markings from (N, M) is denoted by $[N, M\rangle$.

Definition 2. A *labeled Petri net* is a tuple $N = (P, T, F, A, l)$ where (P, T, F) is a Petri net, A is a set of activity labels, and $l \in T \rightarrow A$.

Given a marked Petri net (N, M), an *occurrence sequence* is a sequence of transitions that can be fired one after another. Given the set of all possible occurrence sequences, we called to the set of sequences of A, assigned to the transitions by l, the *language of* N, denoted by $\mathcal{L}(N)$. In general, the models obtained are restricted to a subclass called workflow nets (WF-nets).

Definition 3. Let $N = (P, T, F, A, l)$ be a labeled Petri net and $\bar{t}$ a transition not in $P \cup T$. N is a *workflow net* (WF-net) if and only if: a) P contains an input place i (source place) such that $\bullet\, i = \emptyset$, b) P contains an input place o (source place) such that $o \bullet = \emptyset$, c)$\bar{N} = (P, T \cup \{\bar{t}\}, F \cup \{(o, \bar{t}), (\bar{t}, i)\}, A \cup \{\tau\}, l \cup \{(\bar{t}, \tau)\})$ is strongly connected.

Definition 4. Let $N = (P, T, F, A, l)$ be a WF-net with an input place i and an output place o. N is *sound* if and only if: 1) for any marking $M \in [N, [i]\rangle$, $o \in M$ implies $M = [o]$, 2) for any marking $M \in [N, [i]\rangle$, $[o] \in [N, M\rangle$, 3) $(N, [i])$ contains no dead transitions.

B. Process Discovery and Trace Clustering

A process discovery method can be stated as follows.

Definition 5. A *process discovery algorithm* is a function that maps an event log λ onto a process model M such that M can replay the behavior expressed by the traces in the log: $\lambda \subseteq \mathcal{L}(M)$ [3].

The process model M can be presented in multiple forms, for example, BPMN, EPC, YAWL, Process Trees, or a Petri net. However, in this paper we were assume that the model is a sound WF-net. Moreover, the representativeness of the discovered WF-net is evaluated using four quality measures named *fitness*, *precision*, *generalization,* and *complexity*.

As mentioned, we will focus on the complexity of the model. There is no precise way to define the complexity. However, there is an agreement that a model is more complex the more elements it has, and there are metrics well-defined to measure this.

In the process discovery problem, a complex model is obtained when the log contains very heterogeneous traces. This heterogeneity usually comes from the interleaving of different *variants of a process*, which share some or all activities, but their structure is not necessarily similar.

Trace clustering consists of splitting the log into homogeneous sets of traces, so that the models obtained from these sub-logs are simpler than the model obtained from the whole log (Fig. 1).

Sometimes the same trace belongs to more than one cluster; however, in this paper, a trace will belong to exactly one cluster, i.e., the log is *partitioned* into clusters.

Definition 6. Let L an event log, a *trace clustering* of L is a partition of $L = \{L_1, L_2, \ldots, L_n\}, n \in \mathbb{N}$, i.e. $L_i \cap L_j = \emptyset,\ \forall\, i \neq j$, and $\bigcup L_i = L$.

Of course, the simple act of splitting the log does not lead to simpler models. For the clustering to make sense, it is necessary

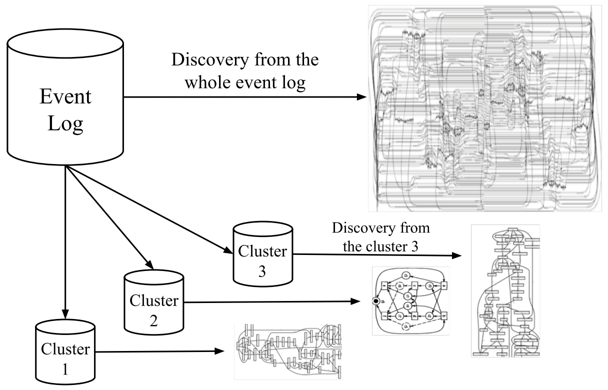

Event Log
Discovery from the whole event log
Cluster 3
Discovery from the cluster 3
Cluster 2
Cluster 1

Fig. 1. Trace clustering approach

that each of the sets L_i, have only traces that are coherent with each other.

C. *Conjoint Occurrence Classes*

A key concept in our proposal is the Conjoint Occurrence Classes (COs), introduced in [8].

Let L and A be a log and its set of activities, respectively. The extraction of the COs is done as follows: for each $a \in A$ and for each $\sigma_i \in L$, the set of activities that occur before the first a and between each pair of $a's$ in σ_i is computed; this is called $Predset(a, \sigma_i)$. Similarly, the set of activities that occur after the last a and between each pair of $a's$ in σ_i is computed; it is called $Succset(a, \sigma_i)$.

Intuitively, $Predset(a, \sigma_i)$ is the set of activities that necessarily must occur in σ_i before activity a occurs. Similarly, $Succset(a, \sigma_i)$ is the set of activities that necessarily occur in σ_i after activity a occurs.

We can compute those activities that appear throughout all the traces of the log in which *a* appears.

For each $a \in A$, the *Occurrence Invariable Set* is computed:

$$Oi(a) = \left(\bigcap_{\sigma_i \in \lambda:\, a \in \sigma_i} Predset(a, \sigma_i) \right) \cup \left(\bigcap_{\sigma_i \in \lambda: a \in \sigma_i} Succset(a, \sigma_i) \right)$$

Each of the components of the previous expression gives us an insight of the structure of the Petri net around a. Then, $Oi(a)$ contains all the activities that occur in a trace each time the activity a occurs. Based on *Oi* the COs are defined.

Definition 7. Let $C \subseteq A$, be a set of activities; it is said that C satisfies the *Conjoint Occurrence Property* if the following condition is fulfilled:

$$\forall\, x, y \in C,\ \ x \in Oi(y) \wedge y \in Oi(x)$$

A set that satisfies this property contains activities that occur if and only if the rest of the activities in the set occur. The maximal sets that satisfy the property are useful to find the largest substructures in the discovered model.

Definition 8. The *Conjoint Occurrence Classes* is a partition $\mathcal{R} \subseteq 2^A$ of A, such that $\forall\, X \in \mathcal{R}$, X fulfills the Conjoint Occurrence Property, and there is not another coarsest partition of A fulfills this property.

It is possible to find different partitions that fulfill the above definition. However, the critical issue is that the activities in a CO always appear in the same order; moreover, they form a *sequential substructure* in a PN.

Example 1: Let L the event log with three traces, $\sigma_1 = \langle x, a, b, d, e, c, a, b, y \rangle$, $\sigma_2 = \langle x, d, e, a, b, y \rangle$, and $\sigma_3 = \langle x, a, b, d, c, a, b, y \rangle$. The sequential substructures shown in Fig. 2 correspond to the COs of L. These are omponents of the underlying Petri net.

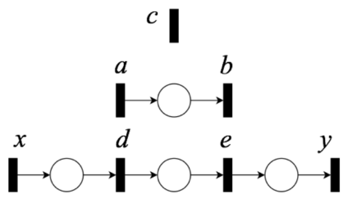


Fig. 2. Substructures issued from COs.

III. Trace Encoding

A. *Overview*

The strategy of the method is described through a toy example. Consider two process models N_1 and N_2 from which two logs of traces are obtained: λ_1={$\sigma_1 = \langle x, c, a, d, b, y \rangle$, $\sigma_2 = \langle x, a, c, d, b, y \rangle$, $\sigma_3 = \langle x, a, b, c, d, y \rangle$ } from N_1 and λ_2={ $\sigma_4 = \langle y, b, a, d, c, x \rangle$, $\sigma_5 = \langle y, d, c, b, a, x \rangle$, $\sigma_6 = \langle y, d, b, a, c, x \rangle$ } from N_2. The sequential substructures that are obtained (using COs) from each log are π_1= $\{x \to c \to d \to y,\ a \to b\}$ and π_2=$\{y \to d \to c \to x,\ b \to a\}$ respectively.

Consider now a trace $\sigma = \langle x, a, c, b, d, y \rangle \in \mathcal{L}(N_1) \cup \mathcal{L}(N_2)$. Based on the sequential substructures π_1 and π_2, it is possible to determine whether σ belongs to $\mathcal{L}(N_1)$ or $\mathcal{L}(N_2)$; in this case, $\sigma \in \mathcal{L}(N_1)$.

In highly flexible environments, traces of different process variants are mixed. For this reason, the COs extracted from the whole log are very small. In our example, if traces from N_1 and N_2 were mixed, each activity would be a CO. If each activity is a CO, then there is no way of knowing which process variant a specific trace belongs to. The larger the size of the COs, the more chance to correctly separate the traces of the different process variants.

The proposed method clusters the traces, by striving for the COs obtained in each cluster to be as large as possible. To achieve that, a *partial view* of the log is obtained from the perspective of one single activity (for each activity). With each partial view, it is possible to calculate the similarity between traces considering only one activity. Afterward, the traces are clustered using such a similarity.

The fact that two traces belong to the same cluster with respect to one activity indicates that the set of invariant occurrences extracted from these two traces will be larger than if the traces belonged to different clusters. It is reasonable to expect that the traces of the same process variant are similar in all or almost all activities. For this reason, the final clusters are formed by similar traces with respect to more activities.

B. *Clustering method*

The proposed method consists of five steps, which are described below. We use the log λ in Table I (**Example 2**).

TABLE I. Event-Traces

id	Traces	id	Traces
σ_1	$\langle x, c, d, a, b, y \rangle$	σ_7	$\langle y, d, c, b, a, x \rangle$
σ_2	$\langle x, a, c, d, b, y \rangle$	σ_8	$\langle y, b, d, c, a, x \rangle$
σ_3	$\langle x, a, b, c, d, y \rangle$	σ_9	$\langle y, b, a, d, c, x \rangle$
σ_4	$\langle x, c, a, d, b, y \rangle$	σ_{10}	$\langle y, d, b, c, a, x \rangle$
σ_5	$\langle x, c, a, b, d, y \rangle$	σ_{11}	$\langle y, d, b, a, c, x \rangle$
σ_6	$\langle x, a, c, b, d, y \rangle$	σ_{12}	$\langle y, b, d, a, c, x \rangle$

1) *Pred/Succset extraction*

In the first step, for each activity $a \in A$ and each trace $\sigma_i \in \lambda$, the sets $Predset(a, \sigma_i)$ and $Succset(a, \sigma_i)$ are computed. These sets capture the information of one activity within a particular trace. In this example, the $Predset$ and $Succset$ for activity a within a trace σ_1 are $Predset(a, \sigma_1) = \{c, x, d\}$, $Succset(a, \sigma_2) = \{b, y\}$.

2) *Characterization of traces by activities*

In the second step, we embed the traces within a vector space using the $Predset$ and $Succset$ as features. For this, we use an approach similar to "bag of words", that is, a trace will be described from the perspective of an activity a, by those activities that always occur before and those that always occur after the occurrence of each a.

Based on the above notions, we define matrices containing information about one activity throughout the entire log.

Definition 9. For each $a \in A$, a matrix $\mu^a \in \{0,1\}^{|\lambda| \times 2|A|}$ is built as follows:

$$\mu^a_{ij} = \begin{cases} 1 & if\ A_j \in Predset(a, \sigma_i) \\ 1 & if\ \ A_{j-|A|} \in Succset(a, \sigma_i) \\ 0 & other\ case \end{cases} \tag{1}$$

Then, a set of $|A|$ matrices is obtained, each one representing the log from the perspective of a particular activity. In Example 2, the matrix for activity a is:

$$\mu^a = \begin{array}{c} \begin{array}{cccccccccccc} a & b & c & d & x & y & a & b & c & d & x & y \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

3) *Clustering partial views*

The third step consists of applying a clustering algorithm on each of the $|A|$ matrices defined in the second step, obtaining $|A|$ sets of clusters.

It is clear that given an activity $a \in A$ and traces $\sigma_1, \sigma_2 \in \lambda$, the rows associated with these traces in the matrix μ^a, will be closer the more elements in common have the sets $Predset(a, \sigma_1)$ and $Predset(a, \sigma_2)$, and $Succset(a, \sigma_1)$ and $Succset(a, \sigma_2)$, i.e., the more similar they are with respect to the activity a.

Definition 10. For each $a \in A$, and $n \in \mathbb{N}^+$, the set of n clusters $C^a = \{C^a_1, \dots, C^a_n\}$ obtained grouping the rows of μ^a will be called *a-similar sub-logs*. Two traces within an *a-similar sublog* are named *a-similar traces.*

Example. In the example $C^a = \{\{\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6\}, \{\sigma_7, \sigma_8, \sigma_9, \sigma_{10}, \sigma_{11}, \sigma_{12}\}\}$

4) *Trace encoding*

In the fourth step, the sets of clusters obtained in the third step will be used as features for a new embedding. Again, we use an approach like bag of words; that is, a trace will be represented by those clusters to which it belongs.

Definition 11. Let us assign to the set $\mathcal{C} = \bigcup_{a \in A} C^a$ a fixed ordering. The matrix $\boldsymbol{\mu} \in \{0, 1\}^{|L| \times |q|}$, given by:

$$\boldsymbol{\mu_{ij}} = \begin{cases} \mathbf{1} & \sigma_i \in \boldsymbol{C_j} \\ \mathbf{0} & other\ case \end{cases} \tag{2}$$

is called the *COBE matrix*, where $q = \sum_{i=1}^{|A|} |C^{a_i}|$. The rows of this matrix will be the final encodings for the traces.

The COBE matrix for the log of Example 2 is:

$$\boldsymbol{\mu} = \begin{array}{c} \begin{array}{cccccccccccc} C^a_1 & C^a_2 & C^b_1 & C^b_2 & C^c_1 & C^c_2 & C^d_1 & C^d_2 & C^x_1 & C^x_2 & C^y_1 & C^y_2 \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \end{array}$$

5) *Trace Clustering*

In the fifth and last step, a clustering algorithm is applied to matrix $\boldsymbol{\mu}$. It is clear that given $\sigma_\mathrm{i}, \sigma_\mathrm{j} \in \lambda$, the rows associated with these traces in the matrix $\boldsymbol{\mu}$ will be closer the more the rows associated with these traces in the μ matrix will be closer if there are more activities $a \in A$ such that σ_i and σ_j are *a-similar.*

In the example, the clusters found are $Cluster1 = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6\}$ and $Cluster2 = \{\sigma_7, \sigma_8, \sigma_9, \sigma_{10}, \sigma_{11}, \sigma_{12}\}$. Fig. 3 shows the models discovered from the whole log and each of the clusters using ILP-Miner.

IV. Implementation and experiments

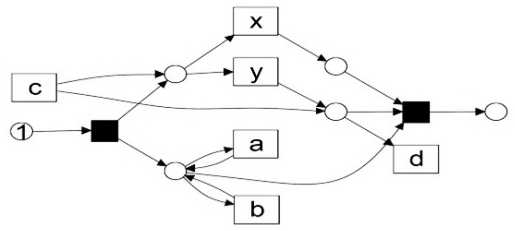
In order to evaluate in practice our proposal, we carried out experiments within two scenarios posed in [9]. The first scenario assesses the ability to identify processes in a log where the traces are mixed. The second one assesses the impact of clustering on the complexity of the discovered model.

A. *Tools and Logs*

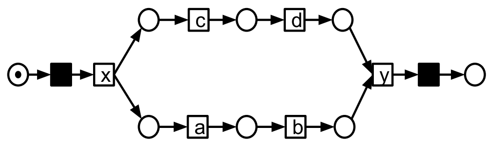
We run experiments with four different event logs, which can be seen in Table II.

TABLE II. Event Logs Characteristics

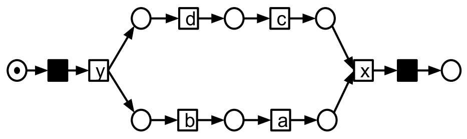
Name	Ref.	#Traces	#Activities
BPI2012	[10]	13,087	36
Sepsis	[11]	1,000	16
Hospital	[12]	100,000	18
Road	[13]	150,370	11



a)



b)



c)

Fig. 3. Discovered net a) from the whole log; b) from the cluster1; c) from the cluster2.

Tests were performed using the K-means algorithm; the implementations available in Sklearn [14] were used. For obtaining the WF-net, we use the implementation of the Inductive Miner [15] available in ProM [16]. The results were compared with those obtained from clustering the embeddings derived using Trace2Vec [6].

B. Scenario 1: Discrimination of Processes

As mentioned before, the objective of the first scenario is to test the discrimination capacity of the proposal. For this, a log was created, merging subsets of multiple logs. An effective trace clustering method should produce the clusters corresponding to the real logs.

For this experiment, we used three event logs, Hospital, Road, and Sepsis. Five hundred traces were randomly selected from each of these logs, and the activities were anonymized, resulting in a log of 1,500 traces and 16 activities. Since we are using k-means, the number of clusters must be specified beforehand. In this case, since the number of merged processes is known, we set it to 3.

In Fig. 4, we can see the *adjusted rand score* (ARS) and the *normalized mutual information score* (NMIS), of the clusters obtained through our proposal and using Trace2Vec with different embedding sizes (32, 64, 128). The NMIS reflects the correlation between the predicted and the real class. On the other hand, ARS represents the percentage of correct decisions, that is, the percentage of traces assigned to your real process.

The results showed a similar performance with both methods. NMIS is slightly better with COBE, however Trace2Vec with vector size 64 scores highest in ARS. Although it may seem that a score below 70% is not a good result, these are satisfactory considering that the number of samples is relatively small.

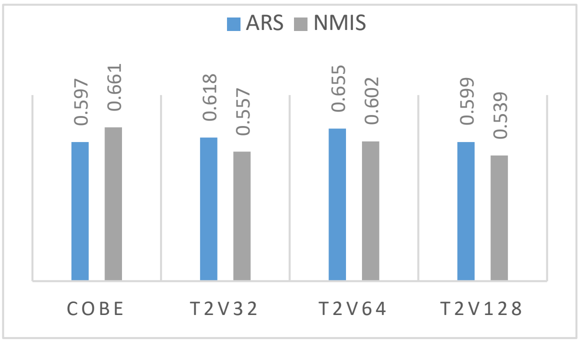


Fig. 4. Scores obtained by COBE and T2V with size 32, 64, and 128.

C. Scenario 2: Model Complexity Reduction

The second scenario aims to evaluate the quality of the models discovered from the clusters obtained. As we mentioned earlier, the quality criterion for comparing the models is complexity. An effective trace clustering method should lead to simpler models.

For this scenario, three metrics were used [17]:

- *Extended Cardoso Metric* (ECaM): This metric locally measures the syntactic complexity of the model. Each place *p* is penalized for the number of subsets of places accessible from it.
- *Extended Cyclomatic Metric* (ECyM): This metric measures the complexity of the modeled behavior. Since it is based on the reachability graph, it is only applicable if the model is bounded (such as in sound WF-net).
- *Structuredness Metric* (SM): This metric "tries to capture the complexity of the model as it is perceived by humans". [17] SM seeks to measure both syntax and behavior; in addition it considers the interaction between different elements (it is not local).

These metrics are defined for WF-nets; thus, a requirement for the discovery algorithm used in this experiment is that it guarantees to obtain a WF-net. There are several methods, for example, the α–algorithm [18], CoMiner [8], Inductive Miner [15]. We have chosen the Inductive Miner for its capability to deal with actual large logs.

Since the number of clusters is missing, we carry out tests with different numbers (3, 5, 7). Table III shows the minimum, the average, and the maximum values of each *metric/#clusters* combination. In this scenario, all experiments performed with Trace2Vec used embeddings size 64. Moreover, the log used was the whole BPI2012.

For all metrics, a lower value implies a simpler model. The experiment shows that the best results are obtained using our proposal for almost every case, except using 7 clusters with respect to the metric SM. In particular, when using 5 clusters, the simplest models are obtained.

The metrics are not comparable to each other. However, the relationship between behavior and structure seems to be more consistent in COBE, i.e., the ratio between ECyM and ECaM is smaller with the clusters found by COBE.

TABLE III. EXPERIMENTAL RESULTS

		COBE			Trace2Vec		
		3	*5*	*7*	*3*	*5*	*7*
ECaM	Min	25	9	9	57	44	15
	Ave	**50.6**	**33.8**	**37.6**	64.3	56	49.85
	Max	80	80	62	78	78	89
ECyM	Min	120	9	9	152	92	15
	Ave	**522**	**268.2**	**418.1**	3502.6	2209.4	1914.6
	Max	869	869	1312	9937	9977	9816
SM	Min	114	22	22	4016	1301.5	76
	Ave	**7295.5**	**4442.6**	15349.3	20532.8	13877.9	**10224.1**
	Max	15090	15090	53480	14450	43132.5	23872.5

In general, the range of values for the metrics (max - min) is very wide in both methods. These results indicate high variability between the clusters; that is, one cluster can lead to a very simple model and another cluster to one not so much.

V. CONCLUSIONS

We have presented a trace clustering method focused on grouping syntactically similar traces allowing discovering processes models structurally simples. This vector-based encoding approach is supported by activities conjoint occurrence classes allowing gathering traces related to the same underlying process structure. The performed tests have shown that the discovered models are significantly simpler than the approaches encoding semantics of the traces.

An important feature of our method is that the clusters are easily explicable regarding the model from the conjoint occurrence classes and their associated substructures. Usually, in most trace clustering methods, the meaning of the clusters is unclear.

Nevertheless, there are some concerns to address in the current and future research for improving the current proposal. One of these concerns is the dimensionality of the encoding; the dimension depends on the number of activities, which can grow to a few hundred. In these cases, it would be advisable to use some dimensionality reduction method or a vertical clustering that reduces the number of activities.

Finally, although some operators of CoMiner method were used, which is focused on obtaining WF-nets, the proposal is not limited to simplifying this type of model. It is necessary to evaluate the effectiveness of COBE with other classes of models, using a more comprehensive range of discovery algorithms and metrics suitable for each type of model.

REFERENCES

[1] W. M. P. van der Aalst et al., "Process mining manifesto" in Business Process Management Workshops, Berlin, Germany:Springer, pp. 169-194, 2012.

[2] Song M., Günther C.W., van der Aalst W.M.P. (2009) “Trace Clustering in Process Mining”. In: Ardagna D., Mecella M., Yang J. (eds) Business Process Management Workshops. BPM 2008.

[3] van der Aalst WMP (2016) Process mining: data science in action. Springer, Heidelberg

[4] R. Xu and D. C. Wunsch-II, Clustering. Wiley, John & Sons, Inc., 2008.

[5] R.P. Jagadeesh Chandra Bose and W.M.P. van der Aalst, "Context Aware Trace Clustering: Towards Improving Process Mining Results", Proc. SIAM Int'l Conf. Data Mining (SDM), pp. 401-412, 2009.

[6] P. De Koninck, S. vanden Broucke and J. De Weerdt, "act2vec trace2vec log2vec and model2vec: Representation Learning for Business Processes" in BPM, Springer, pp. 305-321, 2018.

[7] H. Bui, T. Vu, T. Nguyen, T. Nguyen and Q. Ha, "A Compact Trace Representation Using Deep Neural Networks for Process Mining," 2019 11th International Conference on Knowledge and Systems Engineering (KSE), 2019, pp. 1-5, doi: 10.1109/KSE.2019.8919355.

[8] T. Tapia-Flores, E. Rodríguez-Pérez and E. López-Mellado, "Discovering process models from incomplete event logs using conjoint occurrence classes", Proc. ATAED@ Petri Nets/ACSD, pp. 31-46, 2016.

[9] T. Thaler, S. F. Ternis, P. Fettke and P. Loos, "A comparative analysis of process instance cluster techniques", Wirtschaftsin-formatik, vol. 2015, pp. 423-437, 2015.

[10] BPI challenge 2012. Dataset. doi:10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f

[11] Mannhardt, Felix (2016): Sepsis Cases - Event Log. 4TU.ResearchData. Dataset. https://doi.org/10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460

[12] Mannhardt, Felix (2017): Hospital Billing - Event Log. 4TU.ResearchData.Dataset. https://doi.org/10.4121/uuid:76c46b83-c930-4798-a1c9-4be94dfeb741

[13] de Leoni, M. (Massimiliano); Mannhardt, Felix (2015): Road Traffic Fine Management Process. 4TU.ResearchData. Dataset. https://doi.org/10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5

[14] F. Pedregosa et al., "Scikit-learn: Machine learning in Python", J. Mach. Learn. Res., vol. 12, pp. 2825-2830, Feb. 2011.

[15] S. J. J. Leemans, D. Fahland and W. M. P. van der Aalst, "Discovering block-structured process models from event logs containing infrequent behaviour", Proc. Int. Bus. Process Manage. Workshops, pp. 66-78, 2013.

[16] H.M.W. Verbeek, J.C.A.M. Buijs, B.F. van Dongen, and W.M.P. van der Aalst. XES, XESame, and ProM 6. In P. Soffer and E. Proper, editors, Information Systems Evolution, volume 72 of Lecture Notes in Business Information Processing, pages 60-75. Springer-Verlag, Berlin, 2010.

[17] K.B. Lassen and W.M.P. van der Aalst, "Complexity Metrics for Workflow Nets", Information and Software Technology, vol. 51, no. 3, pp. 610-626, 2009.

[18] W.M.P. van der Aalst, A.J.M.M. Weijters and L. Maruster, "Workflow Mining: Discovering Process Models from Event Logs", IEEE Trans. Knowledge and Data Eng., no. 9, pp. 1128-1142, Sept. 2004