Hindawi
Mathematical Problems in Engineering
Volume 2024, Article ID 7033048, 9 pages
https://doi.org/10.1155/2024/7033048

*Research Article*

# Splitting Long Event Sequences Drawn from Cyclic Processes for Discovering Workflow Nets

**Yolanda Alvarez-Pérez and Ernesto López-Mellado**

*CINVESTAV Unidad Guadalajara, Av. Del Bosque 1145, Col. El Bajio, Zapopan 45019, Jalisco, Mexico*

Correspondence should be addressed to Ernesto López-Mellado; e.lopez@cinvestav.mx

Received 6 September 2023; Revised 27 November 2023; Accepted 13 December 2023; Published 3 January 2024

Academic Editor: Carlos-Renato Vázquez



This paper addresses the preprocessing of event sequences issued from cyclic discrete event processes, which perform activities continuously whose delimitation of jobs or cases is not explicit. The sequences include several occurrences of the same events due to the iterative behaviour, such that discovery methods conceived for workflow nets (WFN) cannot process such sequences. In order to handle this issue, a novel technique for splitting a set of long event traces $S = \{S_k\}$ ($|S| \geq 1$) exhibiting the behaviour of cyclic processes is presented. The aim of this technique is to obtain from $S$ a log $\lambda = \{\sigma_i\}$ of event traces representing the same behaviour, which can be processed by methods that discover WFN. The procedures derived from this technique have polynomial-time complexity.

## 1. Introduction

In discrete event processes, modelling is essential for designing management or control systems or analysing processes in operation. In the latter case, automated modelling of discrete-event processes from the recorded system behaviour is a valuable resource for process reengineering. In the areas of business process and manufacturing systems, automated modelling is an active research matter; in the first area, it is called process discovery [1], while in the second one it is named process identification [2].

*1.1. Automated Modelling.* In both areas, the aim is to build discrete-event models from records of event data generated by the processes; such event data are captured in the form of event sequences or traces, which reveal the actual process behaviour. The models must represent clearly sequential and concurrent behaviours; finite automata and Petri nets (PNs) are the formalisms mostly used.

The source of event traces, called the event log, is the management information systems [3–6] or the process controllers [2, 7]. In each type of process, the logs are represented in different formats. In business processes, the event logs are composed of large multisets of traces; each trace describes a process execution called a case. In manufacturing processes, the activities are continuously performed iteratively; the delimitation of jobs or cases is not explicit. Thus, the event logs are composed of a few (usually one) very long sequences.

*1.2. Event Log Preprocessing.* Extracting the iterative subsequences from long task sequences is a way to isolate the executions of $t$-components of the workflow net (WFN) to discover, allowing splitting the long sequences into multiple traces. This approach allows the application of diverse techniques that discover WFN to event logs drawn from the manufacturing processes.

Existing discovery methods for WFN cannot always process long sequences from cyclic process, in particular, when initial events occur again in the sequence due to the iterative behaviour of the process; the obtained models are less readable or, in some cases, wrong. Consider the log $S = \{abcdabcecd\}$; the discovered model obtained using a standing method [8] is shown in Figure 1(a). Conversely, when the single sequence of the log is split into $\lambda = \{abcd, abcecd\}$, the same discovery method yields the WFN in Figure 1(b); the extended WFN replays $S$.

Splitting or partitioning an event log is a strategy held for several purposes: trace clustering [9, 10], reduction of the

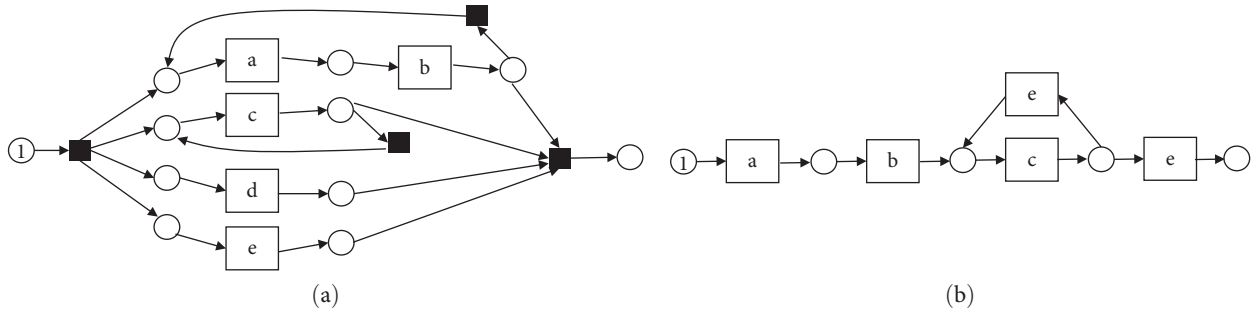



Figure 1: Models discovered from $S$ (a) and $\lambda$ (b).

surplus language for fault diagnosis [11], model simplification [12], discovering unobservable behaviour [13], and model refinement [14]. Methods dealing with the problem of sequence segmentation for improving the translation from Japanese to English have been proposed [15, 16].

*1.3. Contribution.* In this paper, a novel technique for splitting long task sequences issued from highly repetitive cyclic processes into subsequences is proposed. To the best of our knowledge, there are no methods addressing the stated problem. The method processes a reduced set of long event traces $S = \{S_k\}$ ($|S| \geq 1$) and obtains a log $\lambda = \{\sigma_i\}$ of event traces representing the same behaviour. The purpose of this processing is to apply WFN discovery algorithms, in particular, those dealing with the silent transitions.

The paper is organised as follows: Section 2 presents the notation on PNs, WFNs, and the splitting problem; Section 3 describes the splitting trace method; Section 4 presents the implementation and tests; finally, Section 5 presents the conclusions.

## 2. Background and Problem Statement

This section presents the basic concepts and notation of ordinary PNs and WFNs used in this paper. For further details the reader can consult to the study by van der Aalst et al. [1]. Additionally, the sequence splitting problem is formulated.

*2.1. Petri Nets*

*Definition 1.* An ordinary PN structure $G$ is a bipartite digraph represented by the three-tuple $G = (P, T, F)$; where: $P = \{p_1, p_2, \ldots, p_{|P|}\}$ and $T = \{t_1, t_2, \ldots, t_{|T|}\}$ are finite sets of nodes named places and transitions, respectively; $F \subseteq P \times T \cup T \times P$ is a relation representing the arcs between the nodes.

For any node $x \in P \cup T$, ${}^\bullet x = \{y | (y, x) \in F\}$ and $x^\bullet = \{y | (x, y) \in F\}$. The incidence matrix of $G$ is $C = [c_{ij}]$; where $c_{ij} = -1$ if $(p_i, t_j) \in F$ and $(t_j, p_i) \notin F$; $c_{ij} = 1$ if $(t_j, p_i) \in F$ and $(p_i, t_j) \notin F$; $c_{ij} = 0$ otherwise.

The places in $P$ can be empty or marked by one or more tokens. A marking $M$: $P \to \mathbb{N}$ determines the number of tokens within the places; where $\mathbb{N}$ is the set of nonnegative integers. A marking $M$, usually denoted by a vector $(\mathbb{N})^{|P|}$, describes the current state of the modelled system.

*Definition 2.* A Petri net system or Petri net (PN) is the pair $N = (G, M_0)$, where $G$ is a PN structure and $M_0$ is an initial marking. $R(G, M_0)$ denotes the set of all reachable markings from $M_0$.

*Definition 3.* A PN system is 1-*bounded* or *safe* iff, for any $M_i \in R(G, M_0)$ and any $p \in P$, $M_i(p) \leq 1$. A PN system is *live* iff, for every reachable marking $M_i \in R(G, M_0)$ and $t \in T$ there is a $M_k \in R(G, M_i)$ such that $t$ is enabled in $M_k$.

*Definition 4.* A *t-invariant* $Y_i$ *of a* PN is a nonnegative integer solution to the equation $CY_i = 0$. The *support* of $Y_i$ (*t-support*) denoted as $<Y_i>$ is the set of transitions whose corresponding elements in $Y_i$ are positive. $Y$ is *minimal* if its support is not included in the support of other $t$-invariant. A *t-component* $G(Y_i)$ is a subnet of PN induced by a $<Y_i>$: $G(Y_i) = (P_i, T_i, F_i)$, where $P_i = {}^\bullet<Y_i> \cup <Y_i>^\bullet$, $T_i = <Y_i>$, $F_i = (P_i \times T_i \cup P_i \times T_i) \cap F$.

In a $t$-invariant $Y_i$, if we have initial marking ($M_0$) that enables a $t_i \in <Y_i>$, when $t_i$ is fired, then $M_0$ can be reached again by firing only transitions in $<Y_i>$.

*2.2. Workflow Nets*

*Definition 5.* A *WorkFlow net* (WFN) $N$ is a subclass of PN owning the following properties [1]: (i) it has two special places: $i$ and $o$. Place $i$ is a source place: ${}^\bullet i = \emptyset$, and place $o$ is a sink place: $o^\bullet = \emptyset$. (ii) If a transition $t_e$ is added to PN connecting place $o$ to the place $i$, then the resulting PN (called *extended* WFN) is strongly connected.

*Definition 6.* A WFN $(N, M_0)$ is said to be *sound* iff any marking $M_i \in R(N, M_0)$, $o \in M_i \to M_i = [o]$ and $[o] \in R(N, M_i)$ and $(N, M_0)$ contains no dead transitions. An *extended* WFN sound is live and bounded. A WFN can represent a process behaviour by associating task labels to some transitions.

*Definition 7.* A labelled WFN is a four-tuple $(N, M_0, \Sigma, L)$ where $\Sigma$ is a finite set of tasks labels, and $L$: $T \to \Sigma \cup \{\varepsilon\}$ is the labelling function. Transitions labelled with $\varepsilon$ are called *silent* or *unobservable*, otherwise they are called *observable*. Additionally, $\forall$ $t_i$, $t_j \in T$, $t_i \neq t_j$, if $L(t_i)$, $L(t_j) \in \Sigma$ then $L(t_i) \neq L(t_j)$; i.e., two transitions cannot have the same label from $\Sigma$.

*Definition 8.* Let $\Sigma$ be a finite set of tasks labels; an event log $\lambda$ is a set of traces $\sigma_i = A_1 A_2 \ldots A_k \in \Sigma^*$, $|\sigma_i| = k$, $A_j \in \Sigma$; $1 \leq j \leq k$ refers to the task at position $j$.

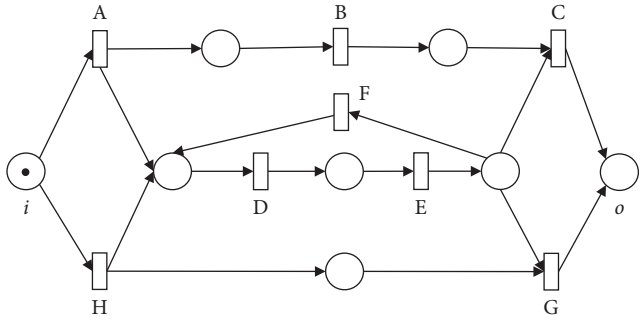



Figure 2: Workflow net of Example 1.

### 2.3. The Problem of Sequence Splitting

*Definition 9.* Given a set of long event traces $S = \{S_k\}$, where $S_k \in T^*$ and $|S| \geq 1$, representing the behaviour of a cyclic discrete even process, the aim is to obtain a set $\lambda = \{\sigma_i\}$ of task traces by splitting the $S_k$, such that the concatenation of traces in $\lambda$ represents the same behaviour expressed in $S$, i.e. an extended WFN discovered from $\lambda$ must replay $S$.

**Assumptions.** *A1. The sequences $S_k$ are arbitrarily long; they capture all the possible actual behaviour of the process. Such sequences are generated by an unknown, live, and 1-bounded cyclic PN. It means that the process is well behaved; there are no deadlocks nor buffer overflows during the recording of traces.*

*A2. In every $S_k$ all the tasks occur at least twice.*

*A3. $S_k$ are recorded from the initial state. Thus, the first tasks are known.*

*Example 1.* Consider the log $S = \{S_1\}$ on $\Sigma = \{A, B, C, D, E, F, G, H\}$, where $S_1$ = HDEGADBEFDECABDECH DEFDEGA DEBFDECHDEGABDECABDEFDECHDEFDEGHDEFDE GADBECADEFBDECHDEGADEB CHDEFDEGADEFDB ECADEFDEBCHDEGHDEG. A suitable splitting technique should determine $\lambda = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, \sigma_7, \sigma_8, \sigma_9, \sigma_{10}, \sigma_{11}\}$, where $\sigma_1$ = ABDEC, $\sigma_2$ = ADBEC, $\sigma_3$ = ADEBC, $\sigma_4$ = ABDE FDEC, $\sigma_5$ = ADBEFDEC, $\sigma_6$ = ADEBFDEC, $\sigma_7$ = ADEFBDEC, $\sigma_8$ = ADEFDBEC, $\sigma_9$ = ADEFDEBC, $\sigma_{10}$ = HDEG, and $\sigma_{11}$ = HDEFDEG that represents the execution of the WFN is depicted in Figure 2. The extended WFN replays $S_1$.

## 3. The Splitting Technique

*3.1. Strategy.* Every $S_k \in S$ is parsed by searching subsequences in $S_k$ that have the same alphabet; such subsequences are represented by a macrotask $\theta_j$, which is replaced in all the $S_k$ that contains this subsequence; this operation is repeated until all the sequences in $S$ are formed only by macrotasks.

*Example 2.* Consider the event log $S$ of Example 1. Then, using the strategy described above, the output of the method is $\lambda = \theta_1 \cup \theta_2 \cup \theta_3 \cup \theta_4$ where $\theta_1 = \{$HDEG$\}$, $\theta_2 = \{$ABDEC, ADBEC, ADEBC$\}$, $\theta_3 = \{$HDEFDEG$\}$, $\theta_4 = \{$ABDEFDEC, ADBEFDEC, ADEBFDEC, ADEFBDEC, ADEFDBEC, ADEFDEBC$\}$. Figure 2 shows the WFN obtained from $\lambda$.

The main steps of the technique are the following. First, an initial splitting of $S_k$, induced by the first task, is performed. Then, the subsequences of $S_k$ are analysed for obtaining the macrotasks $\theta_1$, which are replaced in $S_k$.

*3.2. Basic Operators.* Several operators for handling task traces are introduced below.

*Definition 10.* Let $\lambda$ be an event log over $\Sigma$ and let $a$ be a task in $\Sigma$; for every trace $\sigma_k = x_1 x_2 \ldots x_n \in \lambda$ and $a \in \sigma_k$:

(i) $\tau(x_i, \sigma_k)$ provides the name of the task of position $x_i$ in $\sigma_k$;

(ii) *First*($S'$): gets the first subsequence of the list $S'$;

(iii) $\mathcal{A}$(X) gets the set of tasks (alphabet) used in the object X; $\mathcal{A}(\sigma_k)$ and $\mathcal{A}$ ($\lambda$) gets the set of tasks in a trace $\sigma_k$ and in $\lambda$, respectively.

*Definition 11.* Let $\lambda$ be an event log and $\sigma_k = x_1 x_2 \ldots x_n \in \lambda$ a trace. A macrotask $\theta = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ is a set of traces such that $\mathcal{A}(\sigma_1) = \mathcal{A}(\sigma_2) = \ldots = \mathcal{A}(\sigma_n)$.

Notice that $\mathcal{A}(\sigma_1)$ is the support of a $t$-invariant of the extended WFN to build.

*Definition 12.* Let $S' = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ be a list of subtraces, $\sigma = t_1 t_2 \ldots t_m \in S'$ be a subtrace, $i \in \{1, \ldots, m\text{-}1\}$ and $j \in \{2, \ldots, m\}$ be indexes. Then, the operator *delSet*($S'$, $\sigma$, $\theta$, $i$, $j$) deletes the tasks in $\sigma$ from $i$ to $j$ and replace them with the symbols of the macrotask $\theta$ in $S'$.

*3.3. Splitting Procedures*

*3.3.1. First Splitting.* In the processing of $S_k$, the subsequences to consider are those delimited by a given task symbol $T$ along $S_k$. This search is started using the first symbol of $S_k$; then, a list of sequences $S'$ is formed by all the subsequences of $S_k$ starting with $T$.

The algorithm to split the sequence $S$ in shorter subsequences delimited by the apparition of the first task is presented below.

*Remark.* The computational complexity of Algorithm 1 is $O(|S|)$.

*Example 3.* Consider, the log $S = \{S_1\}$ on $\Sigma = \{A, B, C, D, E, F, G, H, I, J\}$: $S_1$ = ABCHIJDEFG DJABBCABCDEGHIJDEGD JDEFGABCDEFGABBCHIJHIJ obtained from the execution of the model is depicted in Figure 3(a). Then, *splitSeq*($S_1$, $A$) gets $S'$ = {ABCHIJDEFGDJ, ABBC, ABCDEGHIJDEGDJDE FG, ABCDEFG, ABBCHIJHIJ}.

*3.3.2. Determining Macrotasks.* Afterward, the subtrace $\sigma_1$ of $S'$ with the smallest alphabet is chosen and added to the macrotask $\theta_1$; such a subtrace is replaced by $\theta_1$ in $S'$.

Based on $\mathcal{A}(\sigma_1)$ in $\theta_1$, the remainder subtraces $\sigma_r$ that have the same alphabet can be found and then added to $\theta_1$.

Input: $S$, $T$ //The log $S$ and the first task $T$.
Output: $S'$ // A list of the sub-sequences whose first task is $T$
1. $\sigma \leftarrow \emptyset$; $S' \leftarrow \emptyset$;
2. $\forall\ t_i \in S$:
3. If $t_i \neq T$ then:
4. $\sigma \leftarrow \sigma \bullet t_i$
5. else If $i \neq 1$ then:
6. $S' \leftarrow S' \cup \{\sigma\}$; // $\sigma$ is appended to $S'$
7. $\sigma \leftarrow T$;
8. $S' \leftarrow S' \cup \{\sigma\}$
9. Return $S'$

Algorithm 1: FirstSplit.

Input: $S'$, $\sigma$, $\theta$
Output: $S'$, $\theta$
1. $\sigma' \leftarrow \emptyset$; start $\leftarrow 0$; end $\leftarrow 0$; first $\leftarrow 0$;
2. $\forall\ \sigma_i \in S'$:
3. start $\leftarrow 0$
4. $\forall\ t_j \in \sigma_i$: // tracking the symbols of $\sigma_i$
5. If $t_j \in \mathcal{A}(\sigma)$ then
6. If first $= 0$ then
7. start $\leftarrow j$; first $\leftarrow 1$;
8. $\sigma' \leftarrow \sigma' \bullet t_j$
9. else
10. If $\mathcal{A}(\sigma') = \mathcal{A}(\sigma)$: //All the tasks in $\mathcal{A}(\sigma)$ are in $\mathcal{A}(\sigma')$.
11. end $\leftarrow$ j $- 1$
12. $\theta \leftarrow \theta \cup \{\sigma'\}$ //Def. 12. A new sub-trace is appended to the macro-task $\theta$
13. $S' \leftarrow$ delSet($S'$, $\sigma_i$, $\theta$, start, end) //Def. 12
Deletes the tasks in $\sigma_i$ from start to end and replace them with $\theta$ in S′.
14. else: $\sigma' \leftarrow \emptyset$
15. Return S′, $\theta$

Algorithm 2: ReplaceSeq.

The replacing of $\theta_1$ in $S'$ may split the remaining subtraces and then create new subsequences.

This operation is performed again on $S'$ without considering $\theta_1$, then obtaining $\theta_2$, which is included in $S'$ as explained before. In every iteration, new macrotasks $\theta_s$ are created and replaced in $S'$. This process is performed until $S'$ is formed only by macrotasks. The traces in all the macrotasks form the event log.

Now, the procedures (Algorithms 2 and 3) to replace a macrotask $\theta$ in $S'$ and delete the corresponding subsequences are presented below.

*Remark.* The computational complexity of Algorithm 2 is $O(|S'|.|\sigma|)$.

*Example 4.* Consider $S' = \{\sigma_1, \sigma_3, \sigma_3, \sigma_4, \sigma_5\}$, where $\sigma_1 =$ ABC HIJDEFGDJ, $\sigma_2 =$ ABBC; $\sigma_3 =$ ABCDEGHIJDEGDJDEFG; $\sigma_4 =$ ABCDEFG; $\sigma_5 =$ ABBCHIJHIJ and the shortest subtrace $\sigma = \sigma_2$ from Example 3. We replace $\sigma$ with $\theta_1$ in every apparition in $S'$ and split the subsequence where $\sigma$ was replaced. So, we obtain $S' = \{\theta_1$, HIJDEFGDJ, $\theta_1$, $\theta_1$, DEGHIJDEGDJDEFG, $\theta_1$, DEFG, $\theta_1$, HIJHIJ$\}$ and $\theta_1 = \{$ABBC, ABC$\}$.

The procedure below (Algorithm 3) summarises the splitting process.

*Remark.* The computational complexity of Algorithm 3 is $O(|S'|\cdot|\sigma|)$.

*Example 5.* Consider the log $S = \{$ABCHIJDEFGDJABBC ABCDEGHIJDEGDJDEFGABCDEF GABBCHIJHIJ$\}$ from Example 3. We will briefly describe how the splitting technique works.

Input: $S$
Output: $S'$
1. $T \leftarrow \emptyset$; $\sigma \leftarrow \emptyset$; $S' \leftarrow \emptyset$; $i \leftarrow 1$;
2. $T \leftarrow \tau(x_1, S)$; //Def.10 Gets the first task in $S$.
3. $S' \leftarrow FirstSplit(S, T)$;//Alg.1 Splits $S$ in every apparition of $T$.
4. While $\exists\ t_j$ in $\mathscr{A}(S')|\ t_j \in \mathscr{A}(S)$ then
5. $\quad \sigma_{\min} \leftarrow First\ (S')$; // Def.10 Gets the first sub-sequence in $S'$.
6. $\quad \forall\ \sigma \in S'$:
7. $\quad\quad$ If $|\mathscr{A}(\sigma)| < |\mathscr{A}(\sigma_{\min})|$ then
8. $\quad\quad\quad \sigma_{\min} \leftarrow \sigma$;
9. $\quad \theta_i \leftarrow \sigma_{\min}$; // The macro-task is the sub-sequence with the smallest alphabet.
10. $\quad (S', \theta_i) \leftarrow replaceSeq(S', \sigma_{\min}, \theta_i)$; //Alg. 2 Replaces all $\sigma$ in $S'$.
11. $\quad i \leftarrow i + 1$;
12. Return $S'$

Algorithm 3: SplitSequences.

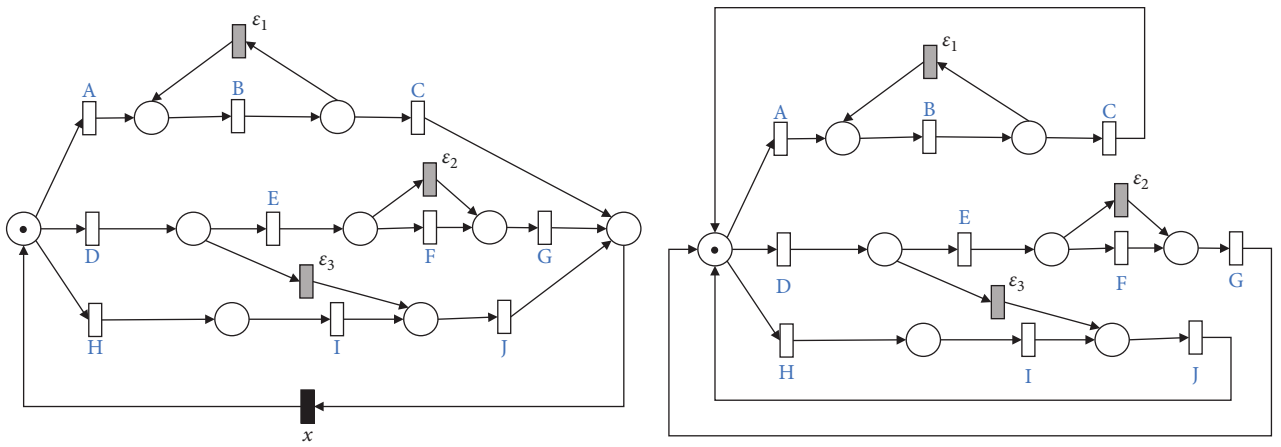



Figure 3: (a) Extended WFN obtained from $\lambda$ and (b) cyclic PN corresponding to $S$.

(1) The first splitting is:
$S' = \{ABCHIJDEFGDJ; ABBC; ABCDEGHIJDEGDJDEFG; ABCDEFG; ABBCHIJHIJ\}$

(2) Then, we get the shortest alphabet subtrace $\sigma = \sigma_2 =$ ABBC; the macrotask $\theta_1 =$ ABBC is created and all the apparitions of the tasks in the alphabet of $\theta_1$ are replaced by the macrotask in $S'$, creating new subtraces and adding the apparitions to $\theta_1$; this is:
$S' = \{\theta_1; HIJDEFGDJ; \theta_1; \theta_1; DEGHIJDEGDJDEFG; \theta_1; DEFG; \theta_1; HIJHIJ\}$, where $\theta_1 = \{ABBC, ABC\}$.

(3) Next, the shortest subtrace is $\sigma =$ DEFG; then the macrotask $\theta_2 =$ DEFG is created, we replace it in $S'$, yielding:
$S' = \{\theta_1; HIJ; \theta_2; DJ; \theta_1; \theta_1; DEGHIJDEGDJ; \theta_2; \theta_1; \theta_2; \theta_1; HIJHIJ\}$.

(4) Then, the shortest subtrace is $\sigma =$ DJ; then the macrotask $\theta_3 =$ DJ is created, we replace it in $S'$, producing:
$S' = \{\theta_1; HIJ; \theta_2; \theta_3; \theta_1; \theta_1; DEGHIJDEG; \theta_3; \theta_2; \theta_1; \theta_2; \theta_1; HIJHIJ\}$

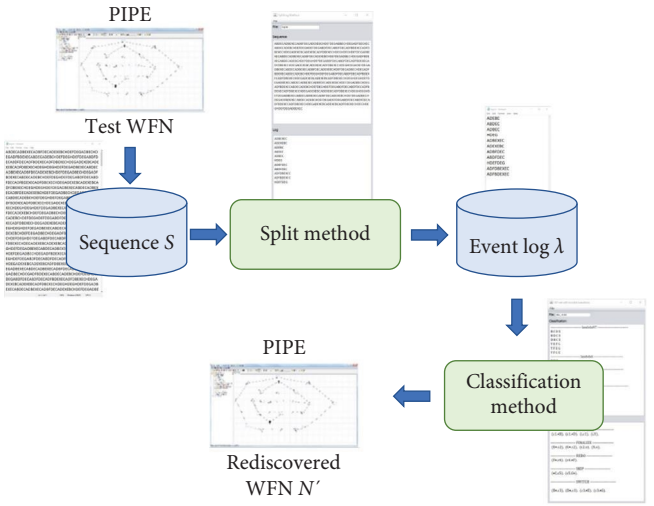



Figure 4: Testing scheme.

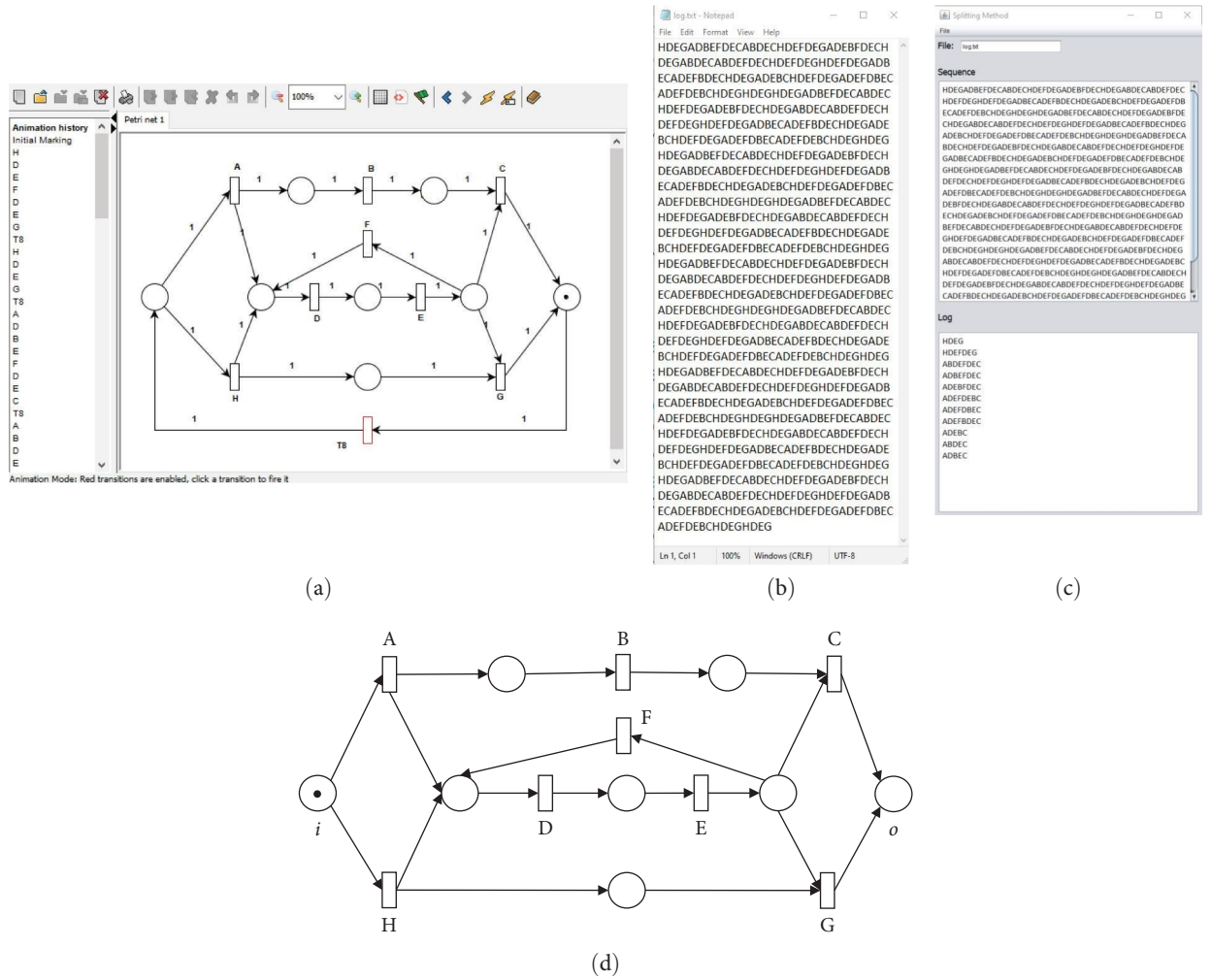



FIGURE 5: Test 1: (a) Extended WFN in PIPE, (b) artificial event log, (c) splitting the sequence, and (d) WFN obtained using the classification method.

(5) Next, the shortest subtrace is $\sigma = \text{HIJ}$ then the macrotask $\theta_4 = \text{HIJ}$ is created; we replace it in $S'$, obtaining:
$S' = \{\theta_1; \theta_4; \theta_2; \theta_3; \theta_1; \theta_1; \text{DEG}; \theta_4; \text{DEG}; \theta_3; \theta_2; \theta_1; \theta_2; \theta_1; \theta_4; \theta_4\}$.

(6) Then, the shortest subtrace is $\sigma = \text{DEG}$; then, the macrotask $\theta_5 = \text{DEG}$ is created; we replace it in $S'$, creating:
$S' = \{\theta_1; \theta_4; \theta_2; \theta_3; \theta_1; \theta_1; \theta_5; \theta_4; \theta_5; \theta_3; \theta_2; \theta_1; \theta_2; \theta_1; \theta_4; \theta_4\}$.

(7) Finally, the set of macrotask is $\{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5\}$, whose subtraces form the event log $\lambda = \{\text{ABBC, ABC, DEFG, DJ, HIJ, DEG}\}$, which is replayed by the WFN (without the transition $x$) shown in Figure 3(a). This WFN is easily transformed into the cyclic PN shown in Figure 3(b).

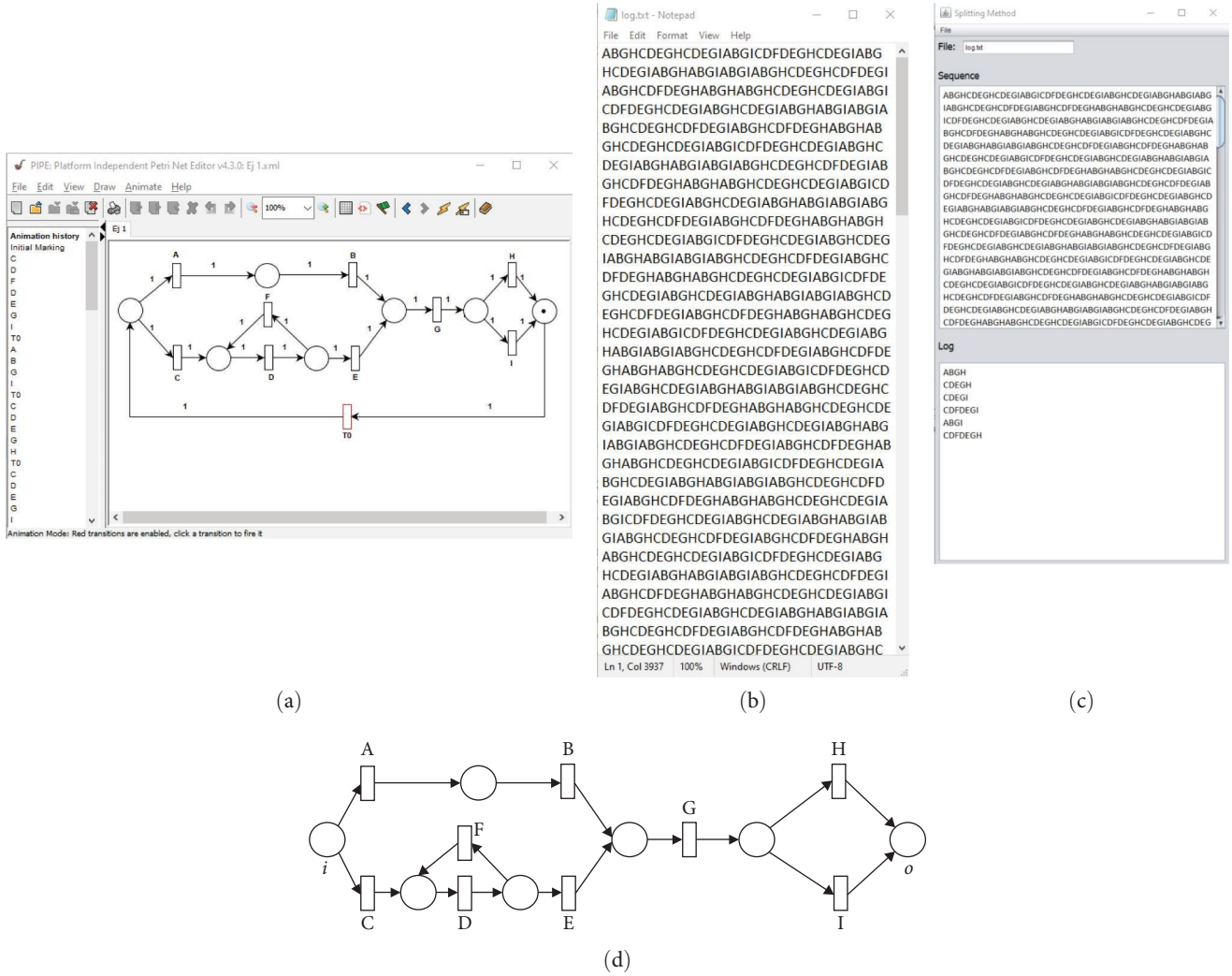
**Property.** *Algorithm 3 processes efficiently an event sequence $S$ yielding a set $S'$ which contains subsequences corresponding to the segmentation of $S$.*

*Proof.* The procedure builds iteratively $S'$ and converges toward a set including only macrotasks. The concatenation of the subsequences represented by the macrotasks in the order they are obtained yields the sequence $S$. Since all the involved algorithms are polynomial-time, the processing is efficient. □

## 4. Implementation and Tests

The algorithms to split a long trace into several traces have been implemented as a software tool. Besides to test the software over sequences and verify the correct splitting, an extended test scheme, described below, is defined.

*4.1. Testing Scheme.* The correctness of the splitting procedure is verified in a controlled manner through a rediscovery scheme, using *artificial event logs*, which are generated as follows. First, a known extended WFN that may contain silent transitions is created and executed in the PN editor PIPE [17]; this WFN contains a transition $t_e$ that allows the cyclic behaviour in the net to get long sequences. Then, the obtained string is processed to delete the apparition of the task $t_e$ in the log and silent transitions labelled with $\varepsilon$. Finally,




(a) (b) (c) (d)

Figure 6: Test 2: (a) Extended WFN in PIPE, (b) artificial event log, (c) splitting the sequence, and (d) WFN obtained using classification method.

the long string is saved in a text file, which is the input of the implemented method.

The developed tool processes the text file that contains the long sequence and splits it into several traces, which are saved in a text file; such traces represent the behaviour of the initial WFN. This text file can be used as input to a discovery process technique [18] to obtain a WFN, which is compared to that used to generate the log. The discovered WFN is an XML file, which can be drawn by PIPE. The followed test scheme is shown in Figure 4.

*4.2. Experiments.* Several case studies using WFN with different structure and size were conducted using the software tool. The following examples are more significant due to their structure rather than the size.

*4.2.1. Test 1.* An execution of the software tool is presented in Figure 5. In Figure 5(a), the extended WFN edited in PIPE is shown; the artificial log is drawn from such a net. The artificial log composed by one sequence of length 1,045 is shown in Figure 5(b). In Figure 5(c), the split log with 11 traces obtained by the execution of the implemented tool is displayed. Then, the WFN discovered by applying the classification method to the split log is displayed in Figure 5(d).

*4.2.2. Test 2.* A second test is presented in Figure 6. In Figure 6(a), the extended WFN is shown. The artificial log with length of 3,937 is shown in Figure 6(b). In Figure 6(c), the obtained log with six traces as result of the execution of the implemented tool is displayed. The WFN obtained using the split log and the classification method is displayed in Figure 6(d).

*4.2.3. Test 3.* In Figure 7, a third test is presented. In Figure 7(a), the extended WFN is shown. The artificial log with length of 10,093 is shown in Figure 7(b). In Figure 7(c), the obtained log with eight traces as result of the execution of the implemented tool is displayed. The WFN obtained using the split log and the classification method is displayed in Figure 7(d).

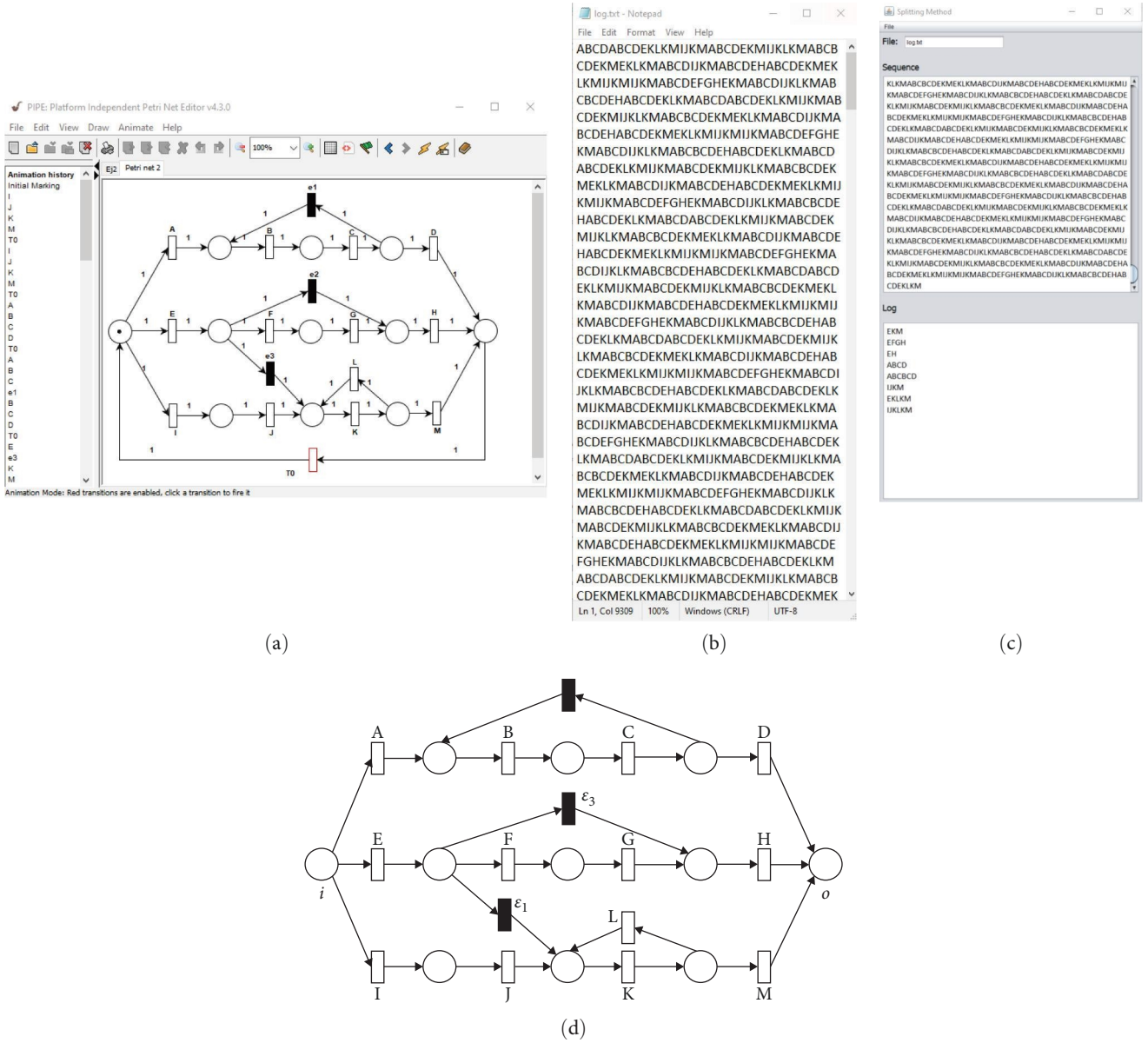



Figure 7: Test 3: (a) Extended WFN in PIPE, (b) artificial event log, (c) splitting the sequence, and (d) WFN obtained using classification method.

## 5. Conclusions

A technique for splitting long event sequences exhibiting the behaviour of cyclic processes has been presented. The result of the processing is an event log from which a WFN can be discovered. Long event sequences are drawn from highly repetitive processes, such as automated manufacturing systems where the initial state is known, but the delimitation of jobs or cases is not specified.

Although, there are discovery methods that deal with the sequences of cyclic processes, this preprocessing technique allows applying many discovery algorithms that build WFN, particularly those that deal with silent transitions [18–20]. In this paper, the method in [18] has been used in the tests to rediscover the models that generate the long sequences.

The event logs obtained from the splitting technique contain traces capturing silent behaviour represented in the discovered WFN by silent transitions of types skip, redo, switch, and finalise. However, these traces cannot always lead to discover initialise silent transitions; it is a pending research.

## Data Availability

No underlying data were collected or produced in this study.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## Acknowledgments

Author, Yolanda Alvarez-Pérez is supported by the CONACYT, Mexico. Ph.D. Grant No. 778009.

## References

[1] W. van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: discovering process models from event logs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 9, pp. 1128–1142, 2004.

[2] A. P. Estrada-Vargas, E. López-Mellado, and J.-J. Lesage, "A black-box identification method for automated discrete-event systems," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 3, pp. 1321–1336, 2017.

[3] A. K. A. de Medeiros, B. F. van Dongen, W. M. P. van der Aalst, and A. J. M. M. Weijters, *Process Mining: Extending the α-algorithm to Mine Short Loops*, Technische Universiteit Eindhoven, 2004.

[4] J. Li, D. Liu, and B. Yang, "Process mining: extending α-algorithm to mine duplicate tasks in process logs," in *Advances in Web and Network Technologies, and Information Management: APWeb WAIM 2007*, K. C.-C. Chang, W. Wang, and L. Chen, et al., Eds., vol. 4537 of *Lecture Notes in Computer Science*, pp. 396–407, Springer, Berlin, Heidelberg, 2007.

[5] L. Wen, W. M. P. van der Aalst, J. Wang, and J. Sun, "Mining process models with non-free-choice constructs," *Data Mining and Knowledge Discovery*, vol. 15, pp. 145–180, 2007.

[6] D. Wang, J. Ge, H. Hu, B. Luo, and L. Huang, "Discovering process models from event multiset," *Expert Systems with Applications*, vol. 39, no. 15, pp. 11970–11978, 2012.

[7] M. Dotoli, M. P. Fanti, A. M. Mangini, and W. Ukovich, "Identification of the unobservable behaviour of industrial automation systems by petri nets," *Control Engineering Practice*, vol. 19, no. 9, pp. 958–966, 2011.

[8] S. J. Leemans, D. Fahland, and W. M. Van Der Aalst, "Discovering block-structured process models from event logs containing infrequent behaviour," in *BPM 2013 International Workshop*, pp. 66–78, Springer international publishing, Beijing, China, 2014.

[9] R. P. Jagadeesh Chandra Bose and W. M. Van der Aalst, "Abstractions in process mining: a taxonomy of patterns," in *Business Process Management BPM 2009*, U. Dayal, J. Eder, J. Koehler, and H. A. Reijers, Eds., Lecture Notes in Computer Science, pp. 159–175, Springer, Berlin, Heidelberg, 2009.

[10] J. De Weerdt, S. vanden Broucke, J. Vanthienen, and B. Baesens, "Active trace clustering for improved process discovery," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 12, pp. 2708–2720, 2013.

[11] S. Schneider and L. Litz, "Automatic partitioning of DES models for distributed fault diagnosis purposes," *IFAC Proceedings Volumes*, vol. 47, no. 2, pp. 21–26, 2014.

[12] J. Li, R. J. C. Bose, and W. M. van der Aalst, "Mining context-dependent and interactive business process maps using execution patterns," in *Business Process Management Workshops: BPM 2010*, M. zur Muehlen and J. Su, Eds., vol. 66 of *Lecture Notes in Business Information Processing*, pp. 109–121, Springer, Berlin, Heidelberg, 2011.

[13] J. Saives, G. Faraut, and J.-J. Lesage, "Identification of discrete event systems unobservable behaviour by Petri nets using language projections," in *2015 European Control Conference (ECC)*, pp. 464–471, IEEE, Linz, Austria, 2015.

[14] E. López-Mellado and T. Flores-Tapia, "Refining discovered Petri nets by sequencing repetitive components," in *ATAED@ Petri Nets/ACSD*, pp. 131–138, 2017.

[15] C.-L. Goh and E. Sumita, "Splitting long input sentences for phrase-based statistical machine translation," in *Proceedings of the 17th Annual Meeting of the Association for Natural Language Processing*, pp. 802–805, Processing society of Japan, 2011.

[16] G. Bernard, A. Senderovich, and P. Andritsos, "Cut to the trace! process-aware partitioning of long-running cases in customer journey logs," in *Advanced Information Systems Engineering: CAiSE 2021*, M. La Rosa, S. Sadiq, and E. Teniente, Eds., Lecture Notes in Computer science, pp. 519–535, Springer, Cham, 2021.

[17] N. J. Dingle, W. J. Knottenbelt, and T. Suto, "PIPE2: a tool for the performance evaluation of generalised stochastic petri nets," *ACM SIGMETRICS Performance Evaluation Review*, vol. 36, no. 4, pp. 34–39, 2009.

[18] Y. Álvarez-Pérez and E. López-Mellado, "Automated modelling of discrete-event processes. Discovering Petri nets including silent transitions by classifying event traces," *International Journal of Modelling and Simulation*, pp. 1–22, 2023.

[19] L. Wen, J. Wang, W. M. P. van der Aalst, B. Huang, and J. Sun, "Mining process models with prime invisible tasks," *Data & Knowledge Engineering*, vol. 69, no. 10, pp. 999–1021, 2010.

[20] Q. Guo, L. Wen, J. Wang, Z. Yan, and P. S. Yu, "Mining invisible tasks in non-free-choice constructs," in *Business Process Management: BPM 2016*, H. R. Motahari-Nezhad, J. Recker, and M. Weidlich, Eds., pp. 109–125, Springer, Cham, 2015.