International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 10 (2020), pp. 3055-3066
 https://dx.doi.org/10.37624/IJERT/13.10.2020.3055-3066

Process Model Discovery from Unlabeled Event Logs by Using Non-Overlapping Sequential Distinct Event Patterns

Muktikanta Sahu[1*], Gopal Krishna Nayak[2] and Rupaj Kumar Nayak[3]

[1]Research Scholar, Department of Computer Science, IIIT Bhubaneswar (Odisha), India.

[2]Professor, Department of Computer Science, IIIT Bhubaneswar (Odisha), India.

[3]Assistant Professor, Department of Mathematics, IIIT Bhubaneswar (Odisha), India.

*(*Corresponding author's ORCID: 0000-0003-1405-3712)*

Abstract

Most of the present-day process discovery methods group events with the help of case identifiers available in the event logs and use the ordering relations that exist among those events to generate process models. However, this may not be possible in many circumstances when the real-world event logs are unstructured in nature and they do not have case identifiers. In such cases, process discovery becomes challenging as it would be difficult to correlate events belonging to the same process instance. The present paper proposes a two-phase approach where the first phase involves discovering the non-overlapping sequential distinct event patterns from an unlabeled event log and the second phase deals with building a process model by establishing correct combinations of those discovered patterns. The applicability of the proposed approach has been tested by carrying out experiments on available real-world event log data.

Keywords: Process Mining, Process Discovery, Non-overlapping patterns, Business Process Intelligence, Business Process Management.

I. INTRODUCTION

Process mining [1], [2], [3], [4], [5], [6], [7], [8], and [9] is a novel discipline which roots its foundations from data mining and business process intelligence. In the recent decade process mining has attracted many researchers to focus their research in this area. Process mining provides a range of tools which can be used for process improvement as well as extracting data driven insights about business processes. The execution histories of a business process (or process instances) are captured in an event log. Hence, an event log serves as a basic source of information for process mining techniques. Further, the information gathered from an event log is primarily used for process discovery [1], which generates a process model for a particular business process.

Majority of the process discovery techniques require at least the following three features in an event log: (i) an identifier that indicates the name of the executed activity, (ii) an identifier of the case to which that particular activity belongs, and (iii) the recorded time (or time-stamp) at which the execution has been over for that activity [2]. An event log might also contain other information, e.g., resource (i.e., the one who executes the activity). Thus, an event log is a collection of all the above features for each recorded event. An example of event log is given in Table 1 which consists of 9 events in 3 cases. Process discovery algorithms try to trace a process model by correlating different events based on their case identifiers and then, forming correlated event clusters. As an example, $\langle A, B, C \rangle$, and $\langle A, D, E \rangle$ are the two different traces of execution which can be captured from event logs from Table 1. It can be clearly seen that $\langle A, B, C \rangle$ can be derived from cases 1 and 3 whereas $\langle A, D, E \rangle$ can be derived from case 2. The corresponding business process model is shown in Figure 1 which has been mined by the Disco process mining tool [9].

Table 1. An example of an event log

Caseid	Activity	Timestamp
1	A	01:22
2	A	01:24
1	B	01:26
2	D	01:29
3	A	01:33
3	B	01:37
1	C	01:43
2	E	01:48
3	C	01:52

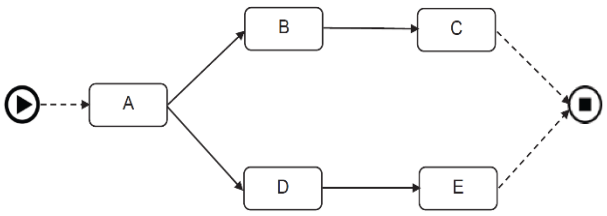

Fig. 1. Generated business process model from the event log given in Table 1.

International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 10 (2020), pp. 3055-3066
 https://dx.doi.org/10.37624/IJERT/13.10.2020.3055-3066

The task of assigning an event to a case for process model discovery is termed as correlation challenge [5]. This task is challenging in applications where there is limited support available from process-aware systems which cannot record case identifiers. Thus, there are circumstances when it is possible to record a large event log without correlating the available events to specific process instances. Typically, in these scenarios the case identifier attribute is missing, and the resulting event log would be a stream of unlabeled events. As consecutive events may be recorded from different instances of processes, it would be difficult to determine whether two events are related or not. In addition to this, also it is unknown that how many process instances are available. This scenario is illustrated in Table 2, which displays the same events from Table 1, but this time the case identifier attribute is not available in the log. Identification of different cases in this event log is not possible and, as a result, it is also not possible to form clusters of related events to form a business process model. Moreover, it is difficult to know which event belongs to which case. Referring to the example event log given in Table 2, both event *A* that is recorded at 01:22 and event *A* that is recorded at 01:24 could be related to *B* that is recorded at 01:37. Certainly, discovering process models in these instances is under-defined. Therefore, the objective of this paper is to develop a method to discover process model from unlabeled event logs or the event logs where the case identifier attribute is missing.

Table 2. An event log example without caseid

Activity	Timestamp
A	01:22
A	01:24
B	01:26
D	01:29
A	01:33
B	01:37
C	01:43
E	01:48
C	01:52

The remaining sections of this paper are organized as follows: Section II gives a glimpse of process mining and covers the literature review relating to the progress made in the field of process discovery. We then present a brief description about event logs and business process models. In the following section, we propose a novel approach to discover process model from an unlabeled event log by using non-overlapping distinct event patterns. Evaluation of the proposed approach and analysis of the same is presented in the following section. The final section contains the conclusion along with future scope of improvement.

II. LITERATURE REVIEW

The fundamental aspects of process discovery are inherited from data mining and business process intelligence. Several process discovery algorithms and the insights about those algorithms can be found in the works described [10], [11], [12], [13], [14], [15], and [16].

The two broad divisions of process discovery algorithms are: "*local*" or "*global*" [39]. Algorithms like Alpha [17], Alpha++ [12], and Heuristic Miner [11] belong to local approaches which rely on the activities present in an event log and the ordering relations that exist among the activities to generate process models using those ordering relations. On the other hand, Genetic [18], [19] and region mining [14], [15] are "global" approaches where a full model is built first, and then successive improvements are done on that model later.

A method to discover process models where caseids are unavailable in the event logs has been proposed by Ferreira and Gillblad [16]. The authors have used a probabilistic approach based on an iterative expectation-maximization approach to discover process models.

Greco [20], Song [21] and [22] in their works proposed trace clustering abstractions to mine complex processes with noise. Very similar to their work, activity level clustering along with appropriate abstractions are proposed by Dongen [2] and Gunther [23], [3] to mine noisy complex processes.

A robust process discovery method has been proposed in [24]. The authors of this work developed a process discovery method where negative examples were infused in an event log to enhance the learning of the process discovery method.

To mine event logs of large size, a divide-and-conquer strategy has been proposed in [25], [26], [27], and [28] by van der Aalst. In this work very large event logs have been first decomposed by adopting a divide and conquer method and then mined.

A heuristic based method to discover process model in case of streaming event data has been presented in [29]. This approach does not include periodic updating of the discovered model. Rather, it is based on rediscovering the model periodically. In this work, the authors have developed a method where a finite queue of events is updated periodically with new events.

Van Eck proposed a state-based process model discovery in [30]. Unlike other process discovery methods which relied on the ordering relations between events in an event log, this method involved identifying different facets, or perspectives for process discovery.

In [31], the authors have tried to localize events by assigning a non-empty set of regions to each event to develop a generic process discovery method. To make a communication or interaction between these regions, shared events have been used.

The challenges in service mining has been reported by van der Aalst in [32] which include the correlation challenge, that we have tried to overcome in this paper.

The authors in their work in [33] proposed a novel technique for process discovery using Maximal Pattern Mining (MPM) to construct patterns based on the whole sequence of events seen

on the traces from an event log having case identifiers. The method ensures soundness of the mined models and can handle loops, duplicate tasks, non-free choice constructs, and long-distance dependencies.

In [34], the authors have proposed a frequent item set mining-based method to extract interesting patterns for knowledge discovery from event logs. But this method considered event logs with case identifiers.

Pourmirza et al in their work in [35] and [36] have presented correlation miner to discover process models from event logs where the case identifiers are missing.

If infrequent behaviors are recorded and considered for process discovery, then we may end up with a complex process model. In [37] and [38] the authors developed filtering methods to remove infrequent behavior from event logs.

The quality of a discovered model is evaluated through several parameters. One such parameter is generalization. In [39] a k-fold cross validation method based on generalization was proposed to evaluate the modeled behavior against the observed behavior of an event log.

The authors in their work in [40] proposed an enhancement over the Heuristic Miner [41]. It was a two-step approach which included discovering accurate but, unstructured and unsound process model in the first step, and filtering out a sound structured model from that unstructured model in the next step.

In [42] the authors presented a method based on autoencoders to improve the quality of process event logs by considering anomalous and missing values of attributes in event logs.

Author of the paper [43] presented a process discovery and how that can be used for effective management of process life cycles to attain ever-growing organizational performance.

At present, a vast literature on process discovery techniques is available which can be tracked from the work presented [44]. Majority of the works listed in [44] either solve the problems like noise present in the event logs, loops in process models, or based on clustering techniques, divide-and-conquer strategies to discover process models. However, the strategy of discovering process models from event logs without case identifiers has been least explored. Except [16], [35], and [36] the process model discovery methods proposed by other researchers either rely on case identifier or any other attribute like case identifier in an event log. However, in many circumstances an event log may be unstructured in nature and it may not have a case identifier field. For example, a firewall log or a web-server log where an event log is a collection of events generated through web page clicks may not have a case identifier field. Thus, extracting a process model from these unstructured logs with the help of present-day process discovery methods may not be suitable. The present paper proposes a novel approach to discover process models from an unlabeled event log by extracting non-overlapping sequential event patterns.

III. EVENT LOGS AND BUSINESS PROCESS MODEL

Several instances of a process may be running simultaneously in a business process management system. Each such instance has an instance number (i.e. caseid), the name of the event or activity being performed, and the timestamp of that activity. These run time execution instances of that process are recorded in an event log and hence, an event log can be treated as a collection of several such process instances or cases [5]. The execution of an activity within a process is synonymous to completion of an event type in a typical business process management system. The event log example shown in Table 1 consists of 9 events over 3 cases. A chronologically ordered sequence of events or activities present in a process instance or case can be called as a trace and that can be symbolically represented as σ. Hence, trace $(\sigma) = t_i(for 1 \le i \le n)$ where t_i is an activity at time point i. The event log which is taken as an example in Table 1 contains the following three traces:

$$\sigma_1 = \sigma_3 = ABC,\ \sigma_2 = ADE$$

However, the above description may not be appropriate for the scenario mentioned in Table 2 as it contains an event log without case identifier. Hence, the redefinition of an event log can be given as follows.

Let α and τ denote the set of activities and timestamps respectively. Thus, an event log L can be defined as $L \subseteq \alpha \times \tau$. ξ_a denotes the set of events for activity $a \in \alpha$, such that $\xi_a = \{(a,t) \mid (a,t) \in L\}$ Similarly, timestamp at which event $e \in \xi_a$ has completed can be represented by t_e.

A process discovery algorithm aims at building a process model and in literature there are several notations available to represent a discovered business process model (e.g. Petri-Nets, BPMN, Workflow net, YAWL) [5]. We have adopted the business process model definition available in Disco [9] process mining tool for our paper.

We denote a business process model as a directed *weighted* graph $G = (\upsilon, \varepsilon, \omega)$ such that:

- $\upsilon = \{(a,n) \mid a \in \alpha \wedge n = \mid \xi_a \mid\}$, where α represents the set of activities and n denotes the frequency of that activity in the event log for each node;
- The set of edges ε can be represented as $\varepsilon = (\upsilon \times \upsilon)$;
- $\omega : \varepsilon \to N$ is a mapping function which assigns natural numbers to the edges, where $\omega(x,y) = n$ if and only if it is found that the activity y directly follows the activity x in an event log n times. Hence, $e \in \varepsilon$ if and only if $\omega(e) > 0$.

IV. NON-OVERLAPPING DISTINCT EVENT PATTERNS FOR PROCESS DISCOVERY

One of the fundamental problems in bioinformatics is pattern discovery which has applications in multiple sequence alignment, protein structure and function prediction, drug target discovery, characterization of protein families, and promoter

signal detection [45]. The primary objective of any pattern discovery approach is to identify an unknown pattern in each set of sequences. Similarly, in process discovery which is one of the fundamental disciplines in business process mining, the primary objective is to build a business process model by extracting event sequence behavior from an event log. But, in most of the cases the challenge is simplified as the case identifiers are available to let us know which events belong to which process instances. Considering the event log given in Table 2, the challenge of discovering a process model can be drafted as a problem of discovering event patterns and building process model from the discovered event patterns.

The challenge of pattern discovery is computationally hard in many cases which means that it cannot be guaranteed that there exists a fast algorithm that would generate the best possible solution to the problem. Thus, many approaches are based on the exhaustive search mechanism. In the worst case these approaches may run in exponential time, but sophisticated pruning techniques can be adopted to make feasible searches for typical input data.

In the simplest form, the working of exhaustive search can be described as follows. In the first step, all possible patterns which satisfy the constraints imposed by the user are enumerated. In the second step, the number of instances of each such pattern in input sequences are calculated. In the third step, a statistically significant score is assigned to each pattern based on the number of instances counted for each pattern. At last, based on a threshold score the patterns which qualify that threshold are generated as output patterns and the rest are discarded.

IV.I. Discovering non-overlapping distinct-event patterns and their frequencies

Discovering non-overlapping event patterns from a given sequence was first proposed by Ding et al. [46]. We have adopted the same method to discover non-overlapping event patterns and followed a few constraints to tackle the challenge of finding relevant event patterns to discover a process model from an event log without case identifiers effectively. The requirements and the formal definition to discover non-overlapping event patterns are given in this section.

Let ξ represents a set of events. An ordered list of events can be denoted by a $sequence$ $S = \langle e_1, e_2, e_3, \ldots, e_{length} \rangle$, where $e_i \in S$ is an event. The sequence S can also be written as: $S = e_1, e_2, e_3, \ldots, e_{length}$. The i^{th} event e_i in the sequence S as $S[i]$.

Definition 1 ***Subsequence and Landmark*****:** Sequence $S = e_1, e_2, e_3, \ldots, e_m$ is a subsequence of another sequence $S' = e_{1'}, e_{2'}, e_{3'}, \ldots, e'_n (m \le n)$ denoted by $S \sqsubseteq S'$ (or S' is a supersequence of S), if there exists a sequence of integers (positions) $1 \le l_1 < l_2 < \ldots < l_m \le l_n$ such that $S[i] = S'[l_i]$ (i.e., $e_i = e'_{l_i}$) for $i = 1, 2, \ldots, m$. Such a sequence of integers $\langle l_1, l_2, \ldots, l_m \rangle$ is called landmark of S in S'. Note: there may be more than one landmark of S in S'. ■

A $pattern$ $P = e_1 e_2 \ldots e_m$ is also a sequence.

Definition 2 ***Instances of Pattern*****:** For a $pattern$ P in a sequence S, if $\langle l_1, l_2, \ldots, l_m \rangle$ is a landmark of pattern $P = e_1 e_2 \ldots e_m$ in S, pair $\langle l_1, l_2, \ldots, l_m \rangle$ is said to be an instance of P in S. ■

$The\ set\ of\ instances\ of\ P\ in\ S$ can be denoted by $S(P)$.

The number of occurrences of a pattern repeating in a sequence can be defined as "support". Hence, support of P, $sup(P)$, the total number of instances of P in S; $sup(P) = |S(P)|$. However, there is a problem with $sup(P)$: multiple instances of a long pattern overlap with each other at many positions in a sequence which leads to over-counting of support for such a long pattern. For example, the pattern $P = ABC \ldots Z$ in sequence $S = AABBCC \ldots ZZ$ has support 2^{26}, but pattern AB only has support $2^2 = 4$.

Thus, counting overlapping instances multiple times in the support value need to be avoided. So first, we need to give a formal definition to overlapping instances as follows.

Definition 3 ***Overlapping Instances*****:** Two instances of pattern $P = e_1 e_2 \ldots e_m$ in sequence $S = e_1 e_2 \ldots e_n$ $(1 \le m \le n)$, $\langle l_1, l_2, \ldots, l_m \rangle$ and $\langle l'_1, l'_2, \ldots, l'_m \rangle$, are overlapping if $\exists 1 \le j \le m : l_j = l'_j$. Equivalently, $\langle l_1, l_2, \ldots, l_m \rangle$ and $\langle l'_1, l'_2, \ldots, l'_m \rangle$, are non-overlapping if $\forall 1 \le j \le m : l_j \ne l'_j$.

■

Example 1: An event sequence S is shown in Table 3. There are 4 landmarks of pattern AB in S. Thus, the 4 instances of the pattern AB in S are: $S(AB) = \{\langle 1,3 \rangle, \langle 1,5 \rangle, \langle 2,3 \rangle, \langle 2,5 \rangle\}$. Instances $(\langle l_1, l_2 \rangle) = (\langle 1,3 \rangle)$ and $(\langle l'_1, l'_2 \rangle) = (\langle 1,5 \rangle)$ are overlapping because $l_1 = l'_1$, i.e., they overlap at the first event $A'(S[1] = A)$. Similarly, the instances $(\langle l_1, l_2 \rangle) = (\langle 1,3 \rangle)$ and $(\langle l'_1, l'_2 \rangle) = (\langle 2,3 \rangle)$ are overlapping as $l_2 = l'_2$. But, the instances $(\langle l_1, l_2 \rangle) = (\langle 1,3 \rangle)$ and $(\langle l'_1, l'_2 \rangle) = (\langle 2,5 \rangle)$ are non-overlapping as $l_1 \ne l'_1$ and $l_2 \ne l'_2$.

International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 10 (2020), pp. 3055-3066
 https://dx.doi.org/10.37624/IJERT/13.10.2020.3055-3066

Table 3. A sample event sequence

Sequence	e_1	e_2	e_3	e_4	e_5	e_6
S	A	A	B	C	B	C

Definition 4 ***Non-redundant Instance Set*****:** A set of instances $I \subseteq S(P)$, of pattern P of sequence S is non-redundant if any two instances in I are non-overlapping. ■

A non-redundant instance set $I \subseteq S(P)$ is maximal if there is no non-redundant instance set I' of pattern P such that $I' \supseteq I$. Thus, the support of pattern P can be defined as the size of a maximal non-redundant instance set $I \subseteq S(P)$. This not only avoids over counting of overlapping instances but also records as many non-overlapping instances as possible for pattern P.

Definition 5 ***Repetitive Support and Support Count*****:** For a non-overlapping pattern P in S, the repetitive support can be defined as $\sup(P) = \max\{|I| \, | \, I \subseteq S(P) \; is \; non-redundant\}$.

The non-redundant instance set I is called a support set of P in S where $|I| = \sup(P)$. ■

Example 2: From Example 1 it can be verified that $|I^{AB}| = 2$ is the maximum size of all possible non-redundant set of I^{AB}. Hence, sup(AB)=2 and I^{AB} is a support set. It is possible to have more than one support set for a pattern. Another possible support set of pattern AB is $I'^{AB} = \{\langle 1,5\rangle, \langle 2,3\rangle\}$.

A formal method to identify the non-overlapping distinct-event patterns and their respective occurrences in an unlabeled event log has been given in Algorithm 1. Algorithm 1 utilizes the preliminaries given in Definition 1, Definition 2, and Definition 3 to tackle the above problem. As events are represented by activity names in an event log, the terms "event" and "activity" are used conversely in this paper.

In Algorithm 1, the following assumptions are considered:

- As the case identifiers are not available in the event log, it is not possible to know the number of process instances completed their execution. By setting the start activity of each pattern P_i as a_f we try to count the number of process instances that have completed execution and their traces recorded in the event sequence S. Hence, in step 2 of Algorithm 1, I_{a_f} is calculated which represents the number of process instances.
- Within a pattern P_i
  (i) each activity is unique and
  (ii) repetition of activities is not allowed.
- For a process there should be at least 2 activities namely "start" and "end" indicating beginning and completion of a process execution. Thus, the minimum length of a pattern P_i is set to 2.

Algorithm 1: Non-overlapping Sequential Distinct Event Patterns

Input: Event sequence (S) from an event log without case identifier

Output: Set of all non-overlapping sequential distinct-event patterns $\xi = \{P_1, P_2, \ldots, P_n\}$

1. Set the very first activity recorded in the event log (denoted by a_f) as the starting activity for all the non-overlapping distinct event patterns that would be discovered. Find out the set of all distinct activities U in the event sequence S .
2. Calculate $I_{a_j} \leftarrow$ The number of occurrence of each activity $a_j \in U$ where $1 \le j \le |U|$. Specifically, calculate $I_{a_f} \leftarrow$ The number of occurrence of a_f
3. Calculate $maxlength \leftarrow |U|$
4. Let $length_{P_i}$ denotes the number of activities in a pattern P_i that would be discovered. Find out the set of all distinct-event patterns $\xi' = \{P_1, P_2, \ldots, P_m\}$ where,
   (i) an event pattern consists of distinct activities,
   (ii) Start activity of each pattern $P_i = a_f$, and
   (iii) $2 \le length_{P_i} \le maxlength$
5. Using Definition 5 calculate $sup(P_i) \leftarrow$ the non-overlapping occurrences of P_i in event sequence S, $\forall 1 \le i \le m$
6. Remove all the patterns P_i from ξ' where, $sup(P_i) = 0$ to find the set of all non-overlapping distinct-event patterns $\xi = \{P_1, P_2, \ldots, P_n\}$ where $1 \le n \le m$.

IV.II. Selecting relevant patterns and constructing business process model

The method proposed in Algorithm 1 produces many patterns with support value greater than 0. Let us assume each individual pattern P_i to be a process instance (or trace) which can be represented as $P_i = \sigma$. Then, some combinations of those individual patterns with their support count can be considered as an event log and hence the event log L can be represented as $L = \{P_1, P_2, \ldots, P_z\}$ where $1 \le z \le n$. In such a case, the following constraints must be obeyed to have a valid event log L and by using which a process model can be

International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 10 (2020), pp. 3055-3066
 https://dx.doi.org/10.37624/IJERT/13.10.2020.3055-3066

generated:

$$\sum_{i=1}^{z} F_{P_i} = I_{a_f} \quad (1)$$

and

$$\sum_{i=1}^{z} F_{a_{j(P_i)}} = I_{a_j} \quad (2)$$

where,

$F_{P_i} = sup(P_i)$,

$F_{a_{j(P_i)}}$ = The number of occurrences of an activity a_j in the pattern P_i,

$a_j \in U$,

$1 \le j \le |U|$ and $1 \le z \le n$.

Example 3: Let us revisit the example event log given in Table 2 to apply the method proposed in Algorithm 1 along with the constraints mentioned in Equation (1) and Equation (2). The results generated on applying Algorithm 1 are:

$a_f = A$

$U = \{A, B, C, D, E\}$

$I_A = 3, I_B = 2, I_C = 2, I_D = 1, I_E = 1$

$I_{a_f} = 3$

$maxlength = 5$

$P_1 = \langle AB \rangle$, $P_2 = \langle AC \rangle$, $P_3 = \langle AD \rangle$, $P_4 = \langle AE \rangle$,
$P_5 = \langle ABC \rangle$, $P_6 = \langle ACB \rangle$, $P_7 = \langle ABD \rangle$,
$P_8 = \langle ADB \rangle$, $P_9 = \langle ACD \rangle$, $P_{10} = \langle ADC \rangle$,
$P_{11} = \langle ABE \rangle$, $P_{12} = \langle AEB \rangle$, $P_{13} = \langle ACE \rangle$,
$P_{14} = \langle AEC \rangle$, $P_{15} = \langle ADE \rangle$, $P_{16} = \langle AED \rangle$,
$P_{17} = \langle ABCD \rangle$, $P_{18} = \langle ABCE \rangle$, $P_{19} = \langle ABDC \rangle$,
$P_{20} = \langle ABEC \rangle$, $P_{21} = \langle ABDE \rangle$, $P_{22} = \langle ABED \rangle$,
$P_{23} = \langle ACBD \rangle$, $P_{24} = \langle ACDB \rangle$, $P_{25} = \langle ACBE \rangle$,
$P_{26} = \langle ACEB \rangle$, $P_{27} = \langle ACDE \rangle$, $P_{28} = \langle ACED \rangle$,
$P_{29} = \langle ADCB \rangle$, $P_{30} = \langle ADBC \rangle$, $P_{31} = \langle ADBE \rangle$,
$P_{32} = \langle ADEB \rangle$, $P_{33} = \langle ADCE \rangle$, $P_{34} = \langle ADEC \rangle$,
$P_{35} = \langle AEBC \rangle$, $P_{36} = \langle AECB \rangle$, $P_{37} = \langle AEBD \rangle$,
$P_{38} = \langle AEDB \rangle$, $P_{39} = \langle AECD \rangle$, $P_{40} = \langle AEDC \rangle$,
$P_{41} = \langle ABCDE \rangle$, $P_{42} = \langle ABCED \rangle$, $P_{43} = \langle ABDCE \rangle$,
$P_{44} = \langle ABDEC \rangle$, $P_{45} = \langle ABECD \rangle$, $P_{46} = \langle ABEDC \rangle$,
$P_{47} = \langle ACBDE \rangle$, $P_{48} = \langle ACBED \rangle$, $P_{49} = \langle ACDBE \rangle$,
$P_{50} = \langle ACDEB \rangle$, $P_{51} = \langle ACEBD \rangle$, $P_{52} = \langle ACEDB \rangle$,
$P_{53} = \langle ADBCE \rangle$, $P_{54} = \langle ADBEC \rangle$, $P_{55} = \langle ADCBE \rangle$,
$P_{56} = \langle ADCEB \rangle$, $P_{57} = \langle ADEBC \rangle$, $P_{58} = \langle ADECB \rangle$,
$P_{59} = \langle AEBCD \rangle$, $P_{60} = \langle AEBDC \rangle$, $P_{61} = \langle AECBD \rangle$,
$P_{62} = \langle AECDB \rangle$, $P_{63} = \langle AEDBC \rangle$, $P_{64} = \langle AEDCB \rangle$.

$\xi' = \{P_1, P_2, \ldots, P_{64}\}$

The support count of each pattern ($sup(P_i)$) has been calculated and the patterns for which $sup(P_i) = 0$ have been removed. Thus, ξ consists of 24 patterns and can be given as: $\xi = \{P_1, P_2, \ldots, P_{24}\}$. These twenty-four patterns and their respective support count have been given in Table 4.

Table 4. Support count of different patterns

Pattern (P_i)	$sup(P_i)$	Pattern (P_i)	$sup(P_i)$
P_1	2	P_{18}	1
P_2	2	P_{19}	1
P_3	1	P_{20}	1
P_4	1	P_{21}	1
P_5	2	P_{30}	1
P_7	1	P_{31}	1
P_8	1	P_{33}	1
P_{10}	1	P_{34}	1
P_{11}	1	P_{43}	1
P_{13}	1	P_{44}	1
P_{14}	1	P_{53}	1
P_{15}	1	P_{54}	1

Now applying the constraints given in Equation (1) and (2) we have:

$F_{P_5} + F_{P_{15}} = 3$ and

$F_{a_{1(P_5)}} + F_{a_{1(P_{15})}} = F_A = 3$

$F_{a_{2(P_5)}} + F_{a_{2(P_{15})}} = F_B = 2$

$F_{a_{3(P_5)}} + F_{a_{3(P_{15})}} = F_C = 2$

$F_{a_{4(P_5)}} + F_{a_{4(P_{15})}} = F_D = 1$

$F_{a_{5(P_5)}} + F_{a_{5(P_{15})}} = F_E = 1$

International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 10 (2020), pp. 3055-3066
 https://dx.doi.org/10.37624/IJERT/13.10.2020.3055-3066

Thus, $L = \{P_5, P_{15}\}$

The $sup(P_5)$ and $sup(P_{15})$ also exactly matches with the traces available in Table 1.

Now, let us assume that the pattern numbers are the case identifiers of different traces that belong to a particular event log. Thus, an event log L can be created with the help of pattern numbers ($i's$) and the respective patterns ($P_i's$). Eventually, the same process model can be generated as the process model given in Figure 1 by using the event log L which contains the patterns (or traces) $P_5 = \langle ABC \rangle$ and $P_{15} = \langle ADE \rangle$ with the help of the Disco process mining tool [16].

Example 4: Let us consider another event log given in Table 5. The process model that can be generated from this event log by using the Disco process mining tool [9] is shown in Figure 2.

Table 5. An example of an event log

Caseid	Activity	Timestamp
1	A	01:22
2	A	01:24
1	C	01:26
2	B	01:29
2	C	01:33

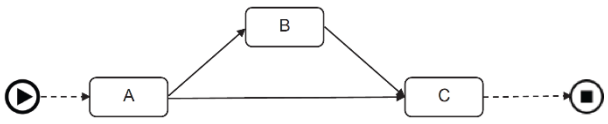

Fig.2. Generated business process model from the event log given in Table 5.

The event log given in Table 5 can be represented as Table 6 without case identifiers.

Table 6. An Event Log without case identifiers of the same event log given in Table 5

Activity	Timestamp
A	01:22
A	01:24
C	01:26
B	01:29
C	01:33

Let us apply the method proposed in Algorithm 1 on the event log given in Table 6. The results generated are:

$a_f = A$

$U = \{A, B, C\}$

$I_A = 2, I_B = 1, I_C = 2$

$I_{a_f} = 2$

$maxlength = 3$

$P_1 = \langle AB \rangle$, $P_2 = \langle AC \rangle$, $P_3 = \langle ABC \rangle$, $P_4 = \langle ACB \rangle$.

$\xi' = \{P_1, P_2, P_3, P_4\}$

The respective support count of each pattern is:

$sup(P_1) = 1$, $sup(P_2) = 2$, $sup(P_3) = 1$, and $sup(P_4) = 1$.

As none of the pattern P_i is having support count $sup(P_i) = 0$, $\xi = \xi' = \{P_1, P_2, P_3, P_4\}$.

Even though we are available with four patterns with support count greater than 0, no combination of these patterns is possible which satisfies the constraints mentioned in Equation (1) and (2). Hence, it is not possible to have an event log L from which a process model can be generated by following the constraints mentioned in Equation (1) and (2). But, if we multiply a weight $w = \frac{1}{2}$ with $sup(P_2)$ and combine this modified support count of P_2 with the support count of P_3, we have:

$\frac{1}{2} \times F_{P_2} + F_{P_3} = 2$ and

$\frac{1}{2} \times F_{a_{1(P_2)}} + F_{a_{1(P_3)}} = F_A = 2$

$F_{a_{2(P_3)}} = F_B = 1$

$F_{a_{3(P_2)}} + F_{a_{3(P_3)}} = F_C = 2$

Thus, $L = \{P_2, P_3\}$ and a process model same as given in Figure 2 can be generated from the event log L.

Things to be noted that the modified support count of P_2 when combined with the support count of P_4 we have the following:

$\frac{1}{2} \times F_{P_2} + F_{P_4} = 2$ and

$\frac{1}{2} \times F_{a_{1(P_2)}} + F_{a_{1(P_4)}} = F_A = 2$

$F_{a_{2(P_4)}} = F_B = 1$

International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 10 (2020), pp. 3055-3066
 https://dx.doi.org/10.37624/IJERT/13.10.2020.3055-3066

$$F_{a_{3(P_2)}} + F_{a_{3(P_4)}} = F_C = 2$$

Thus, $L' = \{P_2, P_4\}$ can be an alternate event log from which another process model as shown in Figure 3 can be generated.

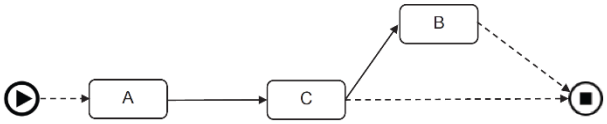

Fig. 3. An alternate business process model of the event log in Table 6.

It can be observed from example 4 that to have a process model we have to consider individual weights for respective patterns. Also, selection of correct weight values is required to avoid generating infinitely many possible process models.

Keeping the above fact in mind, now we formally redefine the constraints mentioned in Equation (1) and (2) as:

$$\sum_{i=1}^{z} w_i \times F_{P_i} = I_{a_f} \quad (3)$$

$$w_i \times F_{P_i} \in Z^+ \quad (4)$$

$$\sum_{i=1}^{z} w_i \times F_{a_{j(P_i)}} = I_{a_j} \quad (5)$$

$$w_i \times F_{a_{j(P_i)}} \in Z^+ \quad (6)$$

where,

$F_{P_i} = sup(P_i)$,

$F_{a_{j(P_i)}}$ = The number of occurrences of an activity a_j in the pattern P_i,

$a_j \in U$,

$0 < w_i \leq 1$, $1 \leq j \leq |U|$, $1 \leq z \leq n$ and Z^+ is the set of positive integers.

Now, the challenge is to find out the correct combination of weights (w_i), so that the constraints mentioned in Equation (3) to (6) are satisfied and we have a relevant set of patterns to form all feasible process models. However, Equation (5) represents a system of linear equations with number of equations as $|U|$ and the number of variables as z with $|U| < z$. Thus, the system of equation reduces to an underdetermined system. Equation (5) can be formulated in a matrix form as $\tilde{A}x = \tilde{b}$, where $\tilde{A}$ is a rectangular matrix whose entries are $F_{a_{j(P_i)}}, \forall 1 \leq j \leq |U|$. x is a column vector whose entries are $w_i, \forall 1 \leq i \leq z$. $\tilde{b}$ is a column vector whose entries are $I_{a_j}, \forall 1 \leq j \leq |U|$. Assuming rank of the matrix $\tilde{A}$ as $|U|$, we can create a submatrix $\tilde{B}$ of size $|U| \times |U|$, formed by $|U|$ linearly independent columns of $\tilde{A}$. We then find the solution by setting $z - |U|$ variables not associated with columns of $\tilde{B}$ to 0 and solving the resulting system of equations $\tilde{B}x_{\tilde{B}} = \tilde{b}$, where $x_{\tilde{B}}$ represents vector associated with the matrix $\tilde{B}$. The solution of the system $\tilde{B}x_{\tilde{B}} = \tilde{b}$ is $x_{\tilde{B}} = \tilde{B}^{-1}\tilde{b}$. Noted that, $x_{\tilde{B}}$ corresponds to the vector representing the columns of the matrix $\tilde{B}$. Thus, the solution is $w_i = (x_{\tilde{B}}, \vec{0})$, which might contain irrelevant values of w_i that may not satisfy the constraint mentioned in Equation (4). Finally, out of all possible solutions we filtered out the relevant values of w_i satisfying the constraint Equation (4).

Referring to the event log given in Table 6 and applying the modified constraints mentioned in Equation (3), (4), (5), and (6) we have the following two sets of weights for the respective patterns P_1, P_2, P_3, and P_4:

$w_1 = 0$, $w_2 = \frac{1}{2}$, $w_3 = 1$, and $w_4 = 0$

and

$w_1 = 0$, $w_2 = \frac{1}{2}$, $w_3 = 0$, and $w_4 = 1$.

Thus, $L = \{P_2, P_3\}$ and $L' = \{P_2, P_4\}$ are the two event logs that can be considered for generating process models. While using L the same process model as given in Figure 2 can be generated, the process model given in Figure 3 can be generated using L'.

It can be clearly seen that for a given event log there can be more than one process model. Hence, out of the several process models, the process model whose *precision* and *recall* measures would be the highest that should be considered as the best process model for the given event log. The *precision* and *recall* can be stated as:

- *precision* = The fraction of cases in the mined model that are correct, i.e., are also in the original business process model, and
- *recall* = The fraction of correct cases that have been found.

Let TP be the types of cases that exist in the mined model and also in the original; FN be the types of cases that do not exist in the mined model but do exist in the original model; and FP be the types of cases that exist in the mined model but do not exist in the original model. Then,

$$precision = \frac{TP}{TP + FP} \quad (7)$$

International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 10 (2020), pp. 3055-3066
 https://dx.doi.org/10.37624/IJERT/13.10.2020.3055-3066

$$recall = \frac{TP}{TP + FN} \quad (8)$$

Table 7. *precison* and *recall* values of L and L'

	$L = \{P_2, P_3\}$	$L' = \{P_2, P_4\}$
precision	1	0.5
recall	1	0.5

As it can be observed from Table 7 that the *precision* and *recall* values of L is higher than L', the model derived from L is the correct model. But, in a real world scenario where the case identifiers are unavailable it may not be possible to find out the *precision* and *recall* values as we may not have an original process model which can be compared with the discovered process models.

V. EVALUATION AND RESULT ANALYSIS

The proposed Algorithm 1 along with the constraints mentioned in Equation (3), (4), (5), and (6) presented in Section IV were evaluated using real-world event logs. Two event logs from BPI challenge 2013 [47] were considered and the corresponding process models were generated by using the Disco [9]. As our algorithm is capable of only dealing with event logs without any loops and the starting activity for all the cases should remain same, we first picked up cases with same starting activity and then removed the cases which were having loops to make the event log acyclic. Further, to check the applicability of our proposed algorithm, all the case identifiers were removed from the modified version of the real-world event log taken from BPI challenge 2013 [47]. The details of the two modified real-world event logs are given in Table 8.

Table 8. Details of the modified event logs

Event log	Unique activities	Events	Cases	Variants
1	5	1380	619	5
2	5	766	357	11

The proposed Algorithm 1 in Section IV was implemented using C++ language and compiled with a GCC compiler version 6.2.0. We developed a MATLAB program using the MATLAB 15(b) package to calculate the weight values w mentioned in the constraints (3) to (6). All the programs were executed in an Intel i5 machine clocked at 2.4 GHz having a memory of 4 GB in a Windows-10 64-bit operating system.

Upon applying Algorithm 1 and the constraints mentioned in Equation (3) to (6) on the modified version of the real-world event logs, we were able to generate several process models. The quality of those generated process models was measured by determining the *precision* and *recall* parameters mentioned Equation (7) and (8) respectively. The best and the worst models in terms of *precision* and *recall* values obtained from event log 1 are shown in Figure 4 and Figure 5 respectively. Table 9 shows the best and the worst *precision* and *recall* measures obtained for the modified real-world event log 1. The model which has the best *precision* and *recall* values was found to be the same as that of the original model (generated for the modified version of the real-world event log 1). But we want to emphasize that in a real-world scenario as the original process model may not be available for comparison, we cannot say which is the best process model out of the generated several process models. In such a case, the views of an expert in that process domain can be considered to pick the best model.

Table 9. *precison* and *recall* values of the best and the worst process models obtained from event log 1 mentioned in Table 8.

	$L = \{P_2, P_3\}$	$L' = \{P_2, P_4\}$
precision	1	1
recall	1	0.8

The actual process model generated from the modified version of the real-world event log 2 is shown in Figure 6 that contains eleven patterns (or cases). As our approach to find the correct combinations of weights (w_i) relies on rank of the matrix representing an underdetermined system of linear equations, we were able to generate process models comprising of five or less patterns for the real-world event log 2 which may not be acceptable in the present context. One of such process models is shown in Figure 7. Thus, it can be concluded that our approach of finding out feasible combination of weights to build a process model works well when the number of patterns available in the actual process model is same as the rank of the matrix representing the underdetermined system of linear equations generated from Equation (5).

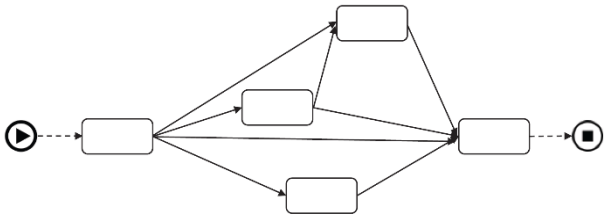

Fig. 4. The best process model obtained from event log 1 given in Table 8.

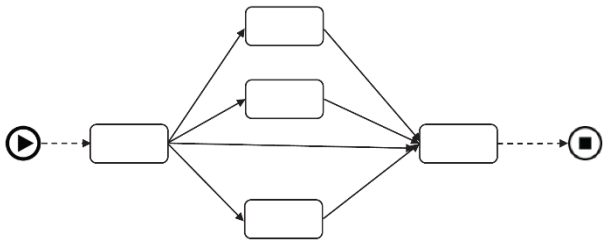

Fig. 5. The worst process model obtained from event log 1 given in Table 8.

International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 10 (2020), pp. 3055-3066
 https://dx.doi.org/10.37624/IJERT/13.10.2020.3055-3066

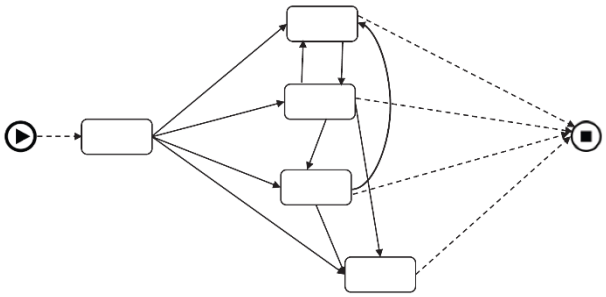

Fig. 6. Actual process model obtained from event log 2 given in Table 8.

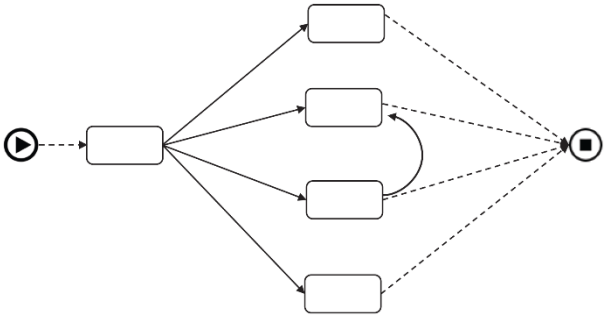

Fig. 7. A process model obtained from event log 2 given in Table 8.

VI. DISCUSSION

We are not the first one to propose a method that deals with the challenge of discovering process models from unlabeled event logs. The methods presented in [16], [35], and [36] also deal with the challenge of discovering process models from unlabeled event logs. In [16], the authors presented a probabilistic approach based on iterative Expectation-Maximization to tackle the said challenge. However, the accuracy of their method relied on the following two factors: the total number of sources in the event log, and on the number of overlapping sources. With high number of sources, it is easier to discover consistent behavior in the event log. But, with increasing number of overlapping sources, it becomes difficult to separate the events belonging to different sources. Our method does not have any such problem with increasing number of overlapping sources.

In [35] and [36], two matrices namely Precede/Succeed matrix and Duration matrix have been constructed and used to create a correlation miner to discover process models from unlabeled event logs. A high value for an entry in the Precede/Succeed matrix indicates that it is more probable to have an edge from the first to the second activity for any two given activities. Similarly, a low value for an entry in the Duration matrix indicates that it is more likely that there is an edge from the first to the second activity for any two given activities. At last, all possible business process models are found out that meet the rule mentioned above and then the best one is selected based on the values from the Precede/Succeed matrix and the Duration matrix. The method relies on the Duration matrix in which an entry indicates the average time difference between events referring to the first activity and events referring to the second activity for any two given activities. If the average time difference between any two activities is too high, then the correlation miner would be unable to correlate two activities. But our proposed method does not rely on the time difference between any two activities.

VII. CONCLUSION AND FUTURE SCOPE

The present work describes a method to discover process model from an event log where the case identifiers are missing. The proposed work relies on discovering the non-overlapping sequential distinct event patterns and their respective frequencies from a given event log without case identifiers believing that the discovered patterns would represent different cases that have taken place in the said event log and by using these cases a process model can be generated. The proposed method has been evaluated by taking the modified versions of the real-world event logs from BPI challenge 2013. Upon applying this method on the modified version of the real-world event logs, we were able to generate several process models as different combinations of discovered event patterns would satisfy the criteria mentioned in Equation (3), (4), (5), and (6). Further, for each discovered process model, the $precision$ and $recall$ values were calculated. The model having highest $precison$ and $recall$ values was considered as the best model for the given even log. However, the present work has the following limitations: (i) it can be applied on an acyclic event log only, (ii) for each and every case available in an event log, the starting activity should remain same, (iii) the discovered patterns were mapped to different cases without any guarantee of whether an activity actually belongs to discovered pattern or not. Thus, the future work shall focus on developing methods which would overcome these issues.

REFERENCES

[1] Aalst, W., Adriansyah, A., Medeiros, A. K. A., Arcieri, F., Baier, T., Blickle, T., Bose, J. C., Brand, P., Brandtjen, R., Buijs, J., et al. (2012). Process mining manifesto. In *Business process management workshops*, pages 169–194. Springer.

[2] Van Dongen, B., Alves de Medeiros, A., and Wen, L. (2009). Process mining: Overview and outlook of petri net discovery algorithms. *Transactions on Petri Nets and Other Models of Concurrency II*, pages 225–242.

[3] Günther, C. and van der Aalst, W. (2007). Fuzzy mining–adaptive process simplification based on multi-perspective metrics. *Business Process Management*, pages 328–343.

[4] Rozinat, A. and Van der Aalst, W. M. (2008). Conformance checking of processes based on monitoring real behavior. *Information Systems*, 33(1):64–95.

[5] Van der Aalst, W. M. (2011). In *Process Mining*. Springer.

[6] van der Aalst, W. M. and Verbeek, H. E. (2008). Process mining in web services: The websphere case. *IEEE Data Eng. Bull.*, 31(3):45–48.

[7] Tiwari, A., Turner, C. J., and Majeed, B. (2008). A review of business process mining: state-of-the- art and future trends. *Business Process Management Journal*, 14(1):5–

22.

[8] Verbeek, H. and van der Aalst, W. M. (2014). Decomposed process mining: The ilp case. In *International Conference on Business Process Management*, pages 264–276. Springer.

[9] Günther, C. W. and Rozinat, A. (2012). Disco: Discover your processes. *BPM (Demos)*, 940:40–44.

[10] Van der Aalst, W., Weijters, T., and Maruster, L. (2004). Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142.

[11] Weijters, A., van Der Aalst, W. M., and De Medeiros, A. A. (2006). Process mining with the heuristics miner-algorithm. *Technische Universiteit Eindhoven, Tech. Rep. WP*, 166:1–34.

[12] Wen, L., van der Aalst, W. M., Wang, J., and Sun, J. (2007). Mining process models with non-free- choice constructs. *Data Mining and Knowledge Discovery*, 15(2):145–180.

[13] Schimm, G. (2004). Mining exact models of concurrent workflows. *Computers in Industry*, 53(3):265–281.

[14] Van Der Aalst, W. M., Rubin, V., Verbeek, H. M., van Dongen, B. F., Kindler, E., and Günther, C. W. (2010). Process mining: a two-step approach to balance between underfitting and overfitting. *Software and Systems Modeling*, 9(1):87–111.

[15] Bergenthum, R., Desel, J., Lorenz, R., and Mauser, S. (2007). Process mining based on regions of languages. In *International Conference on Business Process Management*, pages 375–383. Springer.

[16] Ferreira, D. R. and Gillblad, D. (2009). Discovering process models from unlabelled event logs. In *International Conference on Business Process Management*, pages 143–158. Springer.

[17] Van der Aalst, W. M. and Weijters, A. (2004). Process mining: a research agenda. *Computers in industry*, 53(3):231–244.

[18] de MEDEIROS, A. K., Weijters, A. J., and van der Aalst, W. M. (2007). Genetic process mining: an experimental evaluation. *Data Mining and Knowledge Discovery*, 14(2):245–304.

[19] Turner, C. J., Tiwari, A., and Mehnen, J. (2008). A genetic programming approach to business pro- cess mining. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 1307–1314. ACM.

[20] Greco, G., Guzzo, A., Pontieri, L., and Sacca, D. (2006). Discovering expressive process models by clustering log traces. *IEEE Transactions on Knowledge and Data Engineering*, 18(8):1010–1027.

[21] Song, M., Günther, C. W., and Aalst, W. M. (2009). Trace clustering in process mining. In *Business Process Management Workshops*, pages 109–120. Springer.

[22] Bose, R. J. C. and van der Aalst, W. M. (2009). Context aware trace clustering: Towards improving process mining results. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, pages 401–412. SIAM.

[23] Günther, C. W., Rozinat, A., and Van Der Aalst, W. M. (2009). Activity mining by global trace seg- mentation. In *International Conference on Business Process Management*, pages 128–139. Springer.

[24] Goedertier, S., Martens, D., Vanthienen, J., and Baesens, B. (2009). Robust process discovery with artificial negative events. *Journal of Machine Learning Research*, 10(Jun):1305–1340.

[25] Van Der Aalst, W. M. (2012). Decomposing process mining problems using passages. In *Interna- tional Conference on Application and Theory of Petri Nets and Concurrency*, pages 72–91. Springer.

[26] Van der Aalst, W. M. (2013). Decomposing petri nets for process mining: A generic approach. *Distributed and Parallel Databases*, 31(4):471–507.

[27] Van der Aalst, W. M. (2014). Process mining in the large: a tutorial. In *Business Intelligence*, pages 33–76. Springer.

[28] Van Der Aalst, W. M. (2013). A general divide and conquer approach for process mining. In *Computer Science and Information Systems (FedCSIS), 2013 Federated Conference on*, pages 1–10. IEEE.

[29] Burattin, A., Sperduti, A., and van der Aalst, W. M. (2012). Heuristics miners for streaming event data. *arXiv preprint arXiv:1212.6383*.

[30] van Eck, M. L., Sidorova, N., and van der Aalst, W. M. (2016). Discovering and exploring state-based models for multi-perspective processes. In *International Conference on Business Process Management*, pages 142–157. Springer.

[31] van der Aalst, W. M., Kalenkova, A., Rubin, V., and Verbeek, E. (2015). Process discovery using localized events. In *International Conference on Applications and Theory of Petri Nets and Concurrency*, pages 287–308. Springer.

[32] van der Aalst, W. M. (2013). Challenges in service mining: record, check, discover. In *International Conference on Web Engineering*, pages 1–4. Springer.

[33] Liesaputra, V., Yongchareon, S., and Chaisiri, S. (2016). Efficient process model discovery using maximal pattern mining. In *International Conference on Business Process Management*, pages 441– 456. Springer.

[34] Djenouri, Y., Belhadi, A., and Fournier-Viger, P. (2018). Extracting useful knowledge from event logs: a frequent itemset mining approach. *Knowledge-Based Systems*, 139:132–148.

[35] Pourmirza, S., Dijkman, R., and Grefen, P. (2015). Correlation mining: mining process orches- trations without case identifiers. In *International Conference on Service-Oriented Computing*, pages 237–252. Springer.

[36] Pourmirza, S., Dijkman, R., and Grefen, P. (2017). Correlation miner: Mining business pro- cess models and event correlations without case identifiers. *International Journal of Cooperative Information Systems*, page 1742002.

[37] Conforti, R., La Rosa, M., and ter Hofstede, A. H. (2017). Filtering out infrequent behavior from business process event logs. *IEEE Transactions on Knowledge and Data*

Engineering, 29(2):300–314.

[38] Mannhardt, F., de Leoni, M., Reijers, H. A., and van der Aalst, W. M. (2017). Data-driven process discovery-revealing conditional infrequent behavior from event logs. In *International Conference on Advanced Information Systems Engineering*, pages 545–560. Springer.

[39] Van Dongen, B., Carmona, J., Chatain, T., and Taymouri, F. (2017). Aligning modeled and observed behavior: a compromise between computation complexity and quality. In *International Conference on Advanced Information Systems Engineering*, pages 94–109. Springer.

[40] Augusto, A., Conforti, R., Dumas, M., La Rosa, M., and Bruno, G. (2018). Automated discovery of structured process models from event logs: The discover-and-structure approach. *Data & Knowledge Engineering*, 117:373–392.

[41] Weijters, A. J. and Van der Aalst, W. M. (2003). Rediscovering workflow models from event-based data using little thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162.

[42] Nguyen, H. T. C., Lee, S., Kim, J., Ko, J., and Comuzzi, M. (2019). Autoencoders for improving quality of process event logs. *Expert Systems with Applications*, 131:132–147.

[43] Dymora, P., Koryl, M., and Mazurek, M. (2019). Process discovery in business process management optimization. *Information*, 10(9):270.

[44] Sahu, M. and Nayak, G. K. (2020). Increasing efficiency of process discovery algorithms and process model discovery from unlabeled event logs: A review. *International Journal on Emerging Technologies*, 11(3):383–394.

[45] Brejová, B., Vinar, T., and Li, M. (2003). Pattern discovery. In *Introduction to bioinformatics*, pages 491–521. Springer.

[46] Ding, B., Lo, D., Han, J., and Khoo, S.-C. (2009). Efficient mining of closed repetitive gapped sub- sequences from a sequence database. In *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*, pages 1024–1035. IEEE.

[47] Steeman, W. (2013). Bpi challenge 2013, incidents.