International Journal on Emerging Technologies **11**(3): 383-394(2020)

ISSN No. (Print): 0975-8364
ISSN No. (Online): 2249-3255

Increasing Efficiency of Process Discovery Algorithms and Process model Discovery from Unlabeled Event Logs: A Review

Muktikanta Sahu[1] and Gopal Krishna Nayak[2]
[1]Research Scholar, Department of Computer Science, IIIT Bhubaneswar (Odisha), India.
[2]Professor, Department of Computer Science, IIIT Bhubaneswar (Odisha), India.

(Corresponding author: Muktikanta Sahu)
(Received 18 February 2020, Revised 15 April 2020, Accepted 17 April 2020)
(Published by Research Trend, Website: www.researchtrend.net)

ABSTRACT: Business processes leave behind trails of their execution histories and present-day information systems record these trails in event logs. Process mining helps analysts to have a better insight into these processes by exploiting these event logs. Out of several process mining operations, process discovery is the prominent and most widely researched topic. A process discovery method produces a business process model by correlating events available in an event log. Numerous process discovery techniques have been presented in recent years focusing on issues like the complexity of the generated model, accuracy, and scalability. However, most of these methods incorporate algorithms that are computationally complex and rely on case identifiers to establish the correlation between the events available in an event log to produce a process model. Hence, it becomes important to explore the availability of methods that (i) increase the execution efficiency of the computationally complex process discovery algorithms and (ii) discover process model in the absence of case identifiers. This article not only presents a meticulous review of the methods available for the above-said problems but also highlights the gaps and unexplored facets. There are other review articles available in the domain of process mining, but they do not explicitly focus on the above-said problem statements.

Keywords: Efficiency, Event log, Process discovery, Process mining, Scalability, Review.

Abbreviations: BPMN, Business Process Model Network*;* caseid, case identifier; FHM, Flexible Heuristic Miner; IT, Information Technology; ILP, Integer Linear Programming; IoT, Internet of Things.

I. INTRODUCTION

Modern business processes are event-driven and they rely on information systems to carry out these events. The execution trails of these processes are maintained by information systems in event logs [62]. Process mining techniques not only enable analysts to get a better view of a process but also help the analysts to do an in-depth performance analysis of a business process by using event logs. Thus, event logs play an important role in process mining to derive significant inferences. An event log is a collection of traces of process instances and each trace is an ordered sequence of events related to a particular case.

The rudimentary step involved in process mining is process discovery from event logs and representing the discovered process in a graphical form. A process discovery approach uses an event log as input to produce a process model that represents the control flow relations existing between events (or activities) in the same event log. The effectiveness of such a discovered process model is measured through the accuracy of reflecting the behavior captured in the event log. The discovered process model must not be complex. Also, it should have high *fitness*, *generalization*, and *precision* value [61].

The challenge of discovering process models from event logs is not new. Many researchers have proposed their unique approaches to tackle this challenge or factors affecting this challenge in the past twenty years.

While a healthy number of proposals are available to deal with the challenge of process discovery, most of them rely on event logs having case identifiers (caseid). It becomes pretty easier to correlate events (or activities) belonging to a process trace in the presence of caseids. But, in the absence of caseid's the challenge of process discovery becomes hard to crack. Also, most of the state-of-the-art process discovery techniques are complex and include computation-intensive steps. As modern-day business organizations produce a voluminous amount of data, the event log size increases exponentially. This results in longer execution times of the state-of-the-art process discovery techniques. Thus, it becomes important to explore and review the techniques available for (i) discovering process models that do not rely on caseid's and (ii) increasing the run-time efficiency of process discovery algorithms. This paper aims at presenting a meticulous depiction of the above-said facts.

The remaining portions of the paper are organized as follows. Section II presents the formulation of the searching mechanism for the review process. Section III describes the methods available for process discovery in the presence of caseids and the absence of caseids respectively. Then, the methods which either increase scalability or run-time efficiency of process discovery algorithms are described in Section IV. Section V presents the discussionon the overall findings followed by Section VI which concludes the paper and sketches future work directions.

II. FORMULATING THE SEARCH MECHANISM

To gather the information regarding research done in the field of process discovery, we listed out a set of search strings based on a set of research questions. These search strings were executed on different sources of data. Further, we enforced some criteria for selecting the studies retrieved through the search to be included in the review process.

A. Research Questions

Since the progress made in the field of process mining is now two decades old and many methods are available for process discovery, the objective of this literature review is to accumulate all the methods available for process discovery, segregate and analyze the methods that can discover process model from an unlabeled event log, and list out any such framework that can increase the execution efficiency of process discovery methods. Specifically, our search criteria for the literature review is based on the following three research questions:

[Q1] What are the methods available for process discovery?

[Q2] How to discover process models in case of unlabeled event logs and is there any method available for the same?

[Q3] How to enhance the execution performance of the process discovery methods?

B. Forming the search strings and selection of studies to be included

The core of our search revolves around Q1 (which also includes the proposals targeting Q2) mentioned in the previous section, which aims at tracing out present methods available for process discovery. To achieve this, we considered the following search strings: “process discovery”, “process discovery in process mining”, “process discovery and process mining”, “discovering process models”, “workflow modeling”, “workflow discovery”, and “discovering process models”. We used Google Scholar for searching documents having either of these search strings. Specifically, we retrieved those documents which contained either of these search strings in their title or abstract or keywords. Further, we considered only those documents which purely proposed a new method to discover process model and discarded all the other documents which have proposals on noise reduction or conformance checking in process mining [64]. The studies which target Q3 mentioned in Subsection *A* are also filtered out from the above-mentioned search results. The entire process of searching was over by December 2019.

III. PROCESS DISCOVERY METHODS

Given an event log, discovering a process model from that event log is the primary task in process mining. Process discovery methods are based on machine learning and data mining techniques. The description of several process discovery algorithms along with their respective notations is presented in the following section.

A. Process discovery in the presence of caseid

An important attribute that helps in correlating different events or activities belonging to a particular process instance in an event log is caseid. Within an event log, several caseids might be present, but these must be unique to correlate the events or activities that are executed within a process instance. The majority of the process discovery techniques rely on caseid or an attribute that is very similar to a caseid in an event log to discover process models. We have retrieved and summarized all such techniques or methods that rely on caseid for process discovery below.

Agrawal *et al.*, proposed a method to discover process models from unstructured execution of processes. They applied this technique to workflow management systems. The application of this technique on synthetic data displayed successful evaluation and evolution of existing processes [2].

Cook and Wolf developed a data analysis technique and named it to process discovery where they were able to discover models from event-based data. Specifically, they developed RNet, Ktail, and Markov methods to discover models in the context of software engineering. They used statistical, algorithmic, and probabilistic approaches to develop these methods [13].

Datta proposed the B-F (k,c) algorithm to automatically extract AS-IS process models in the domain of Business Process Reengineering (BPR) and Workflow Management. This algorithmic approach was based on statistics and probability theory [14].

A technique to learn two-component mixture models of global partial orders was proposed by Manilla and Meek [44]. The technique was able to provide an understandable, global view of a set of event sequences. But, the technique lacked many other typical problems like concurrency in the field of process mining. The data mining techniques utilized and proposed a tool named as Process Miner. Process Miner was able to discover exact workflow models from event-based data. But, Process Miner was not robust enough to be applied in a real-life setting [56, 57].

Van Der Aalst *et al.,* [62] proposed an algorithm to discover a process model from a workflow log. They named it α algorithm. The α algorithm was able to mine any workflow represented by a so-called structured workflow net. However, this algorithm was incapable of discovering any arbitrary workflow process as they might have loops. An enhancement over the existing α algorithm was proposed by de Medeiros *et al.,* [3] which was able to overcome the problem of short loops. The authors proposed the α^+-algorithm which was able to correctly mine sound structured workflow nets and handle short loops of length one and two. The α^+-algorithm was implemented in the EMiT tool. Herbst & Karagiannis described the splitpar algorithm, which is part of the InWoLvE framework for process analysis.

This algorithm was based on deriving a so-called stochastic activity graph and converting it into astructured process model. The splitpar algorithm was at par with detecting duplicate activities, but it was incapable of discovering non-local dependencies [32].

vanDongen and van der Aalst presented a process discovery method by aggregating process models obtained from individual procinstances into a Petri net. The intermediate steps in this method were derived by using Event-driven Process Chains (EPCs) [67].

Another process discovery alg ess orithm named Workflow Miner was proposed by Gaaloul *et al*., The elementary dependencies in an event log were modeled through intermediary graphical representations and then the final advanced structural workflow patterns were modeled in this technique [24].

Weijter *et al*., proposed a heuristic-based algorithm to discover a process model from an event log in the process mining domain [77]. The algorithm was named as "HeuristicsMiner" and was able to deal with noise. HeuristicsMiner can express the main behavior recorded in an event log. However, it was unable to represent exceptions present in an event log.

Greco *et al*., proposed a machine learning theory-based approach in [29] and called this technique as DWS mining. Based on a hierarchical and iterative procedure, DWS mining was able to refine the process model in each step as the process model was generated by applying a clustering technique on patterns having similar behavior. This approach not only guaranteed full compliance with the event log but also displayed incremental improvement on the soundness property of the process model.

An induction rule-based approach to predict the causal dependency between activities from a set of event logs having different levels of noise and imbalance was proposed by Mărușter *et al*., [45]. Specifically, a propositional rule induction technique was used to deduce the relations in a preprocessing step.

A combined approach of ILP learning and partial-order planning was proposed by Ferreira and Ferreira to discover process models. By repetitive combined execution of planning and learning, a process model was discovered. The model was represented based on the case data preconditions and effects of its activities [22].

A genetic algorithm-based method was proposed to extract the process model from an event log [15]. The method was successful in dealing the problems like non-trivial constructs and/or noise present in the log. Rather than relying on local information, this genetic algorithm approach used global search techniques to handle these problems. Experiments were carried out using synthetic and real-life logs to show that the fitness measure is complete and precise. The genetic algorithm was embedded in the ProM framework as a plug-in.

Petri net-based discovery methods show their limited capabilities in learning accurate and comprehensible models while dealing with real-world event logs which are highly complex. To overcome such a problem, alternative discovery technique such as the FuzzyMiner was proposed [30]. This technique was more effective as this technique featured better abstraction capabilities.

Wen *et al.,* developed methods to mine non-free-choice constructs as most of the real-life processes display the said behavior. Using Petri-net-based representation, they were able to show two types of causal dependencies between tasks namely implicit and explicit. An algorithm that can deal with these dependencies and also, implemented that algorithm in the ProM Framework. An improvement over the original α algorithm was proposed where they take advantage of both starts and completed event types to detect concurrency. They named their algorithm as the β-algorithm [78].

Lamma *et al*., described the use of ILP to process mining. The search algorithm was guided by the presence of negative sequences in this method. The use of partial-order planning was avoided in this method whilepresentinga userwith an execution plan to accept or reject. Hence, the root of the negative events was not answered immediately in this approach [35].

Greco *et al.,* proposed AWS mining, an enhancement over the preexistingprocess discovery techniques by imposing an abstraction-based method that targeted classification of process models [27]. The method was capable of analyzing different behavior present in an event log in details. A mining algorithmwas combined withan abstraction-based algorithmin this technique to produce a tree-like schema finally. The non-leaf nodes in this tree-like structure represented an abstract process model that further generalized to represent all the different process models in the respective subtree.

Goedertier *et al*., proposed AGNEsMiner which addressed the task of process discovery with the help of first-order classification learning. The event logs were infused with artificially generated negative events (AGNEs). The final output of AGNEsMiner was a Petri net modelrepresenting a process. It was possible to distinguish between the occurrence of either a positive or a negative event from an event log that has been infused with artificially generated negative events through this method. The entire process mining task was based on a classification learning problem. The algorithm was designed in such a way that it can learn the distinguishing conditions that deduce whether an event can happen or not, given an execution history of events of other activities [25].

An enhanced version of the WFMiner was proposed. This enhanced algorithmic technique was capable of dealing with many important process discovery challenges like noise, duplicate tasks, and non-free choice [23].

A method based on Integer Linear Programming (ILP) to discover process models from an event log was proposed. This ILP based method was formerly known as the Parikh language-based region miner which has been derived by using the concepts from the language-based theory of regions an area that belongs to the Petri net domain. The authors claimed that this technique to be useful as this approach allowed for parallelization and was independent of the number of events registered in the event log [65].

To have a balanced trade-off between precise and general process model, a new process discovery technique called FSMMiner/Petrify was proposed. The idea was to represent mined process models through different views at different levels of abstraction. It was a two-step approach where in the first step, a transition system should be constructed from the traces in an event log. The second step involved synthesizing this transition system employing the theory of regions and finally, a Petri net was constructed [52, 63].

Carmona *et al.,* described Genet. This technique was very similar to FSMMiner/Petrify and was able to produce a Petri net from a transition system [11].

To represent the semantics of splits and joins in an event log, a new representation language, and the respective algorithm has been proposed in [76]. The method is based on asplit/join frequency table and named Flexible Heuristic Miner. The proposed algorithm was implemented as a plug-in in the ProM framework and was able to produce easy to understand process models in the case of non-trivial constructs, low structured domains, and the presence of noise.

The very first basic approach to mine declarative process models was proposed [41]. The authors used Declare constraints [49, 80] to extract these models.

An algorithm to extract block-structured Petri nets from event logs was proposed [33]. In a two-step approach, the algorithm first develops an adjacency matrix between all pairs of tasks and then extracts block-structured models consisting of primary sequence, parallel, choice, optional, loop, and self-loop structures by analyzing the available information. The method was developed as a standalone tool and named as HK.

The process discovery method that returns causal nets was proposed by Greco *et al.,* [28, 26]. A causal net is built by capturing the causal relations between activities in an event log. The causal relations are gathered from an event log and encoded in this method. The required background knowledge is captured in terms of precedence constraints of the resulting model topology and then the algorithm is formulated based on reasoning problems over these precedence constraints.

A two-phase approach to discover Declare constraints and named it MINERful [18]. In the first phase of this approach, the occurrences of activities and their interplay in the event log were computed. The second phase dealt with checking the validity of Declare constraints by querying the knowledge base created with the previous statistical data structure.

The Inductive Miner was proposed which was based on the extraction of process trees from an event log [37]. The proposed technique ensured soundness while dealing with infrequent behavior and large event logs during process discovery. The technique has been implemented in ProM. An extension of this work where modeling of cancellation behavior is supported has been published [36].

To discover Declare constraints along with semantics that considers data conditions was first presented [40]. The authors used first-order temporal logic to derive the data-aware semantics of Declare.

The Proximity Miner presented is another method that produces causal nets. In this method, the behavioral relations between the events in an event log are first extracted and then these are enhanced by using inputs from domain experts [82, 81].

A runtime monitoring framework to do log analysis was developed [1]. The framework was capable of extracting process instances and trace out appropriate metrics simultaneously in a single pass through the event logs. Further, these metrics were used to select traces with certain characteristics to be used for the discovery of process models. The authors took the help of directly-follows graphs to express process models.

Another approach for the discovery of Declare constraints has been presented. Also present the Evolutionary Declare Miner that implements thediscovery task using a genetic algorithm [72].

The Evolutionary Tree Miner was introduced. This was a genetic algorithm-based method that enables the user to discover process based on four quality preferences namely: fitness, precision, generalization, and complexity [8, 9].

To discover Petri nets from large event logs, numerical abstract domains were used [10]. The formal properties of the discovered models were guaranteed in this approach. Also, the method ensured that the discovered Petri nets can reproduce every trace in the log which minimally describes the log behavior.

Ferilli proposed the WoMan framework [20] which included learning and refining process models from an event log by discovering first-order logic constraints. The method ensured incremental learning and adapting the models along with the ability to showcase triggers and conditions on the process tasks and efficiency.

Maggi *et al*., presented the Hybrid Miner [42] which can produce a hybrid process model from an event log. A hybrid process model was a hierarchical model consisting of several nodes and each node represents a sub-process. Further, each sub-process was specified in a declarative or procedural way. To represent procedural sub-processes Petri nets were used, whereas to represent declarative sub-processes Declare were used.

A divide-and-conquer algorithm-based process discovery method known as the Constructs Competition Miner (CCM) was proposed [51]. The method was able to discover block-structured processes from event logs which might be having exceptional behavior.

To extract directed acyclic graphs from event logs based on probabilistic models was proposed [73]. The method extensively used the Bayesian belief network which is one of the most common probabilistic models.

The authors of the work proposed an approach for the discovery of hybrid models named as Fusion Miner. Fusion Minor was based on the semantics devoted to obtaining a fully mixed language, where procedural and declarative constructs can be connected [16].

DGEM proposed was a method to discover BPMN models. In the first step of this two-step approach, a hierarchical view on process models was formally specified. In the second step, an evolution strategy was applied to it. The evolution strategy was driven by the diversity of the process model population. The method was very effective in finding the process models that best represent a given event log [47].

Non-free-choice construct and invisible tasks are two critical structures that need to be dealt with in a process model. The challenge of mining invisible tasks involved in non-free-choice constructs was proposed in an algorithm named $\alpha^{\$}$ [31]. The problem was solved in $\alpha^{\$}$ by introducing new ordering relations. The $\alpha^{\$}$ was able to significantly improve the existing process mining techniques and was implemented as a plug-in of ProM.

Liesaputra *et al.,* proposed the Maximal Pattern Mining (MPM) technique to discover process models from event logs. The method was based on capturing patterns of event sequences in an event log and from

these patterns a process model was generated which were represented through causal nets. This MPM technique was able to handle loops (of any length), duplicate tasks, non-free choice constructs, and long-distance dependencies.

Vazquez *et al.*, proposed ProDiGen [74] where they addressed the problem of process discovery through a genetic algorithm. The method used a new fitness function that considered completeness, precision and simplicity, and specific crossover and mutation operators. The discovered models were represented through causal nets.

Declarative process models consider activities of a business process to be atomic/instantaneous events. But, this is not always true as in realistic environments, process activities may not be instantaneous but would have executed over a time interval and passed through a sequence of states of a lifecycle. The method proposed described a discriminative rule mining approach to enable the existing declarative process discovery techniques to analyze business processes having non-atomic activities [6].

The theory of grammatical inference was used to generate Petri nets for process discovery. This was a standalone application and named as RegPFA [7].

Conforti *et al.*, proposed the BPMN Miner [12]. This was an automated discovery method that produced BPMN models. The generated models contained sub-processes, activity markers like multi-instance and loops along with interrupting and non-interrupting boundary events that could handle exception handling.

The authors in their work proposed the CSM Miner. The CSM Miner was able to discover state machines from event logs. This method focused onthe states of the different process perspectives and discovered the relations among them instead of concentrating on the events or activities that were executed in a particular process. These relations were expressed as Composite State Machines. The CSM Miner was also able to provide an interactive visualization of these multi-perspective state-based models [68].

Li *et al.,* proposed a process mining algorithm named as τ. This method leveraged data carried by tokens during the execution of a business process and tracked the state changes in the so-called token logs. This information used to improve the mining efficiency and mining capability of standard process discovery algorithms [38].

The algorithms to extract the control flow, as well as relevant data parameters from a given event log, has been presented [46]. The authors also showed that how conditional partial order graphs can be used to visualize the obtained results from an event log. The method was called as PGminer and implemented as a Workcraft plug-in as a standalone application.

A mining approach that takes the help of standard SQL for querying the log to directly work on relational event data was introduced [58]. The mining procedure became fast as the detection of certain control-flow constraints was removed and database performance technology was incorporated. Also, customization of queries was possible and process perspectives were covered beyond control flow.

The method presented emphasized that activities without any dependencies in an event log can be executed in parallel. Hence, this method was able to discover process models with concurrency without caring about the completeness criteria of the logs. A tool named ProM-D was developed with this method [59].

To discover sound workflow nets from incomplete event logs is a challenging task. An approach to tackle this challenge was introduced [60]. The activities which infer the behaviors not exhibited in the log were identified through the concept of invariant occurrence. The set of such activities was named conjoint occurrence classes and the proposed method was based on this concept.

Augusto *et al.,* proposed a discovery method that combined the following two: (i) a technique to filter the directly-follows graph induced by an event log, (ii) with an approach to identify combinations of split gateways that accurately capture the concurrency, conflict and causal relations between neighbors in the directly-follows graph. The proposed method was able to produce simple process models with low branching complexity. Also, the produced models were having consistently high and balanced fitness, precision, and generalization [4].

vandenBroucke & De Weerdt presented Fodina, a heuristic-based process discovery technique with a strong focus on robustness and flexibility. The authors were able to identify several drawbacks that impact the reliability of previously existing heuristic-based process discovery techniques. The proposed algorithm has better performance in terms of process model quality, adds the ability to mine duplicate tasks, and allows for flexible configuration options [71].

A method to discover causal nets that optimizes the scalability and interpretability of the outputs was proposed [48]. The process that needs to be analyzed was decomposed into set independent stages so that each stage can be mined separately. With the above implementation, the technique was able to maximize modularity by discovering a stage decomposition.

Verbeek *et al.*, proposed a generic divide-and-conquer approach to discover process models from very large event logs [75]. The approach was based on partitioning the event log into several smaller logs and discovering a model from each of those smaller logs. Then all the models discovered from those sublogs were assembled to form the final model. This method was able to reduce overall complexity and produce high-quality models.

Often process discovery methods filter out infrequent paths and activities from an event log by treating them as noise. However, sometimes, removing this infrequent behavior may lead to a loss of significant insights into the process. Hence, not all infrequent behavior should be considered as noise. Mannhardt *et al.*, proposed a Data-aware Heuristic Miner (DHM) [43]. This process discovery method was able to distinguish between infrequent paths and random noise by using classification techniques based on data attributes. Both data- and control-flow of the process were discovered by using this technique. The applicability of the DHM was evaluated on several real-life event logs.

Table 1: Overview of different process discovery techniques in the presence of caseid.

S. No.	Method	Contributor(s)	Year
1.	General DAG	Agrawal *et al.*, [2]	1998
2.	B-F(k,c)-algorithm	Datta [14]	1998
3.	Rnet, Ktail, Markov	Cook and Wolf [13]	1998
4.	Global partial orders	Manilla and Meek [44]	2000
5.	Process Miner	Schimm [56]	2002
6.	α / α^{+}	van der Aalst *et al.*, [62], [3]	2004
7.	InWoLvE—splitpar	Herbst and Karagiannis [32]	2004
8.	Multi-phase Miner	van Dongen and van der Aalst [67]	2005
9.	Workflow Miner	Gaaloul *et al.*, [24]	2005
10.	HeuristicsMiner	Weijters *et al.*, [77]	2006
11.	DWS Mining	Greco *et al.* [29]	2006
12.	Rule-based approach	Măruşter *et al.* [45]	2006
13.	ILP-partial order	Ferreira and Ferreira [22]	2006
14.	Genetic Miner	Alves de Medeiros et al. [15]	2007
15.	Fuzzy Miner	Günther and van der Aalst [30]	2007
16.	α^{++}	Wen *et al.*, [78]	2007
17.	DecMiner	Lamma *et al.*, [35]	2007
18.	AWS Mining	Greco *et al.*, [27]	2008
19.	AGNEsMiner	Goedertier *et al.*, [25]	2009
20.	β (or Tshinguaα)	Wen *et al.*, [79]	2009
21.	Enhanced WFMiner	Folino *et al.*, [23]	2009
22.	ILP Miner(Parikh)	Werf *et al.*, [65]	2009
23.	FSM Miner/Petrify	van der Aalst *et al.*,[63]	2010
24.	FSM Miner/Genet	Carmona *et al.*, [11]	2010
25.	Aim	Carmona, Cortadella [10]	2010
26.	Flexible Heuristic Miner	Weijters *et al.*, [76]	2011
27.	Declare Miner	Maggi *et al.*, [41]	2011
28.	HK	Huang and Kumar [33]	2012
29.	CNMining	Greco *et al.*, [28]	2012
30.	MINERful	Di Ciccio and Mecella [18]	2013
31.	Inductive Miner - Infrequent	Leemans *et al.*, [37]	2013
32.	Data-aware Declare Miner	Maggi *et al.*, [40]	2013
33.	Proximity Miner	Yahya *et al.*, [81]	2013
34.	WoMan	Ferilli [20]	2013
35.	Process Skeletonization	Abe and Kudo [1]	2014
36.	Evolutionary Declare Miner	vandenBroucke *et al.* [72]	2014
37.	Evolutionary Tree Miner	Buijs *et al.*, [8, 9]	2014
38.	Hybrid Miner	Maggi *et al.*, [42]	2014
39.	Competition Miner	Redlich *et al.*, [51]	2014
40.	Directed Acyclic Graphs	Vasilecas *et al.*, [73]	2014
41.	Fusion Miner	De Smedt *et al.*, [16]	2015
42.	DGEM	Molka *et al.*, [47]	2015
43.	ProDiGen	Vazquez *et al.*,[74]	2015
44.	ProM-D	Song *et al.*, [59]	2015
45.	$\alpha^{\$}$	Guo *et al.*, [31]	2016
46.	Maximal Pattern Mining	Liesaputra *et al.*, [39]	2016
47.	Non-Atomic Declare Miner	Bernardi *et al.*, [6]	2016
48.	RegPFA	Breuker *et al.*, [7]	2016
49.	BPMN Miner	Conforti *et al.*, [12]	2016
50.	CSMMiner	van Eck *et al.*, [69]	2016
51.	τ miner	Li *et al.*, [38]	2016
52.	PGminer	Mokhov *et al.*, [46]	2016
53.	SQLMiner	Schönig *et al.*, [58]	2016
54.	CoMiner	Tapia-Flores *et al.*, [60]	2016
55.	Split miner	Augusto *et al.*, [4]	2017
56.	Fodina	vandenBroucke *et al.*, [71]	2017
57.	Stage miner	Nguyen *et al.*, [48]	2017
58.	Decomposed Process Miner	Verbeek, van der Aalst [75]	2017
59.	Data-aware Heuristic Miner	Mannhardt *et al.*, [43]	2017
60.	Discover and Structure	Augusto *et al.*, [5]	2018
61.	HybridILPMiner	van Zelst *et al.*, [70]	2018

Augusto *et al.,* proposed a method of generating a structured (and sound) process model from an event log. They adopted a two-step approach. They first used a well-known heuristic that could discover an accurate but unstructured model. In the second step, they transformed that unstructured model into a structured sound model. This discover-and-structure approach was proven to outperform other existing methods considering complexity and accuracy measures [5].

An improvement over the ILP miner presented, which was based on hybrid variable-based regions. The number of variables used to solve an ILP based problem can be varied through hybrid variable-based regions. It is important to note thatdifferent numbers of variables have a different impact on average computation time for solving an ILP problem [64, 70].

Santhoshkumar *et al.,* presented a process model based analytics for improving service quality in the healthcare domain by unitizing the combined benefits of Big-data and IoT [55].

Referring to our first research question (Q1): "What are the methods available for process discovery?" we have summarized all the sixty-one studies available to date on process discovery using caseid in Table 1. Each entry in Table 1 represents the method that was proposed, followed by the contributors of that work, and the year of publication.

The algorithms listed in Table 1 are capable of producing different types of models like Procedural, Declarative, and Hybrid. Similarly, the adaptation of modeling languages also varies across different discovery techniques. Some of the popular modeling languages are: Petri nets, Declare, Process trees, WoMan, BPMN, Causal nets, State machines, Directed acyclic graphs, and Partial order graphs. We have not explicitly mentioned the type of models being produced and modeling languages used by different discovery algorithms, as our primary focus was on Q2 and Q3 mentioned in Section II. Also, due to this reason only, we have not focused on parameters like accuracy, precision, fitness, generalization, F-score, and noise, which are being used to determine the effectiveness of a process discovery algorithm. Readers are advised to go through [17, 5] to have a better knowledge of these issues. While [5] presents a detailed review and analysis of the different process discovery techniques published in 2012 onward, a review of the works done before 2012 in the field of process discovery is presented in [17].

B. Process discovery in the absence of caseid

In many unstructured business processes such as service-oriented processes, it may not be possible to capture or record caseids for process instances. Hence, the event logs are generated without caseids and termed as unlabeled event logs. In the absence of caseids, it becomes challenging to correlate different events or activities that belong to the same process instance and therefore, the task of process discovery becomes difficult. In the literature of process discovery, only two methods are available that deal with process discovery in the absence of caseids. We have listed out these two methods.

The problem of discovering process models in the absence of a case identifier was first proposed [21]. The authors presented a probabilistic approach to estimate a process model utilizing an iterative Expectation–Maximization procedure. The same procedure has also been used to find the caseid's in unlabeled event logs. The method was able to discover process models with loops.

Pourmirza *et al.,* proposed a method to address the correlation challenge which arises due to the unavailability of caseid's in an event log, and hence, process discovery becomes a challenging task. The method was solved using Integer Linear Programming (ILP) based on precede/succeed matrix and duration matrix calculated from timestamps of event occurrence within an event log. The method was also able to generate the caseid attribute for the unlabeled event log, but it was limited to discover process models without loops.

Table 2: Overview of different process discovery techniques in the absence of caseid.

S. No.	Method	Contributor(s)	Year	Able to discover splits-joins	Able to discover loops	Able to discover caseid	Tested on synthetic logs	Tested on real-world logs
1	EM-approach	Ferreira and Gilblad	2009	Yes	Yes	Yes	Yes	Yes
2	CorrelationMiner	Pourmirza *et al.,*	2017	Yes	No	Yes	Yes	Yes

The details of the above mentioned two proposals are listed in Table 2. Out of the two methods mentioned in Table 2, the algorithm presented [21] was able to produce a model with precision 61% and recall 41%, which are 24% and 22% lower than the results of the algorithm presented [50] respectively when tested over a real-world event log from the BPI Challenge of 2012 [66]. But, the method proposed [21] can process models having cycles or loops unlike the method proposed [50].

Both the methods were able to detect splits-and-joins, which captures the parallelization and branching available in a process model. Also, these methods were able to detect caseids for respective process instances.

IV. METHODS AVAILABLE FOR INCREASING EFFICIENCY OF PROCESS DISCOVERY ALGORITHMS

Modern-day information technology-enabled organizations produce and record a voluminous amount of data related to the business processes they execute in event logs. Hence, it is obvious that the size of the

event logs increases exponentially as the number of events increases linearly. Given the complex nature of the process discovery algorithms and the exponentially-increasing event logs, it becomes important to have efficient implementations of process discovery algorithms so that process discovery becomes faster. In this regard, we have briefly described the frameworks that are proposed in recent years.

Evermann presented a framework based on the map-reduce approach to enhance the execution efficiency of process discovery algorithms assuming the distributed nature of event log data produced by modern information systems [19]. The map-reduce approachwas applied to two of the well-known process discovery algorithms namely the α algorithm [61] and the FlexibleHeuristic Miner [76] which is an extended version of the originally proposed "HeuristicMiner" [77]. The proposed map-reduce framework was highly scalable but lacked in-memory computations.To test the effectiveness of these proposed framework event logs of a very large size are required. Due to unavailability of such large size real-world event logs, all these three methods were tested on artificial event logs of very large size. The authors used an artificial event log of size 80 GB to show the effectiveness of their proposed Map-Reduce framework on the α and the FHM algorithm respectively [19].

Sahu *et al.* [53] proposed a task-parallelism based approach to increase the execution efficiency of the αalgorithm [62].The authors applied the MPI framework over an artificial event log of the size of 1.02 GB to display the increased execution efficiency of the α algorithm. In this work, independent and computation-intensive steps available in the α algorithm were identified and executed in parallel with the help of distributed memory parallelism available in the message passing programming (MPI) model. The execution efficiency of the proposed MPI-based framework was compared against the serial execution of the α algorithm and the enhanced execution performance expressed through the *speedup* [34] factor. They have exploited task parallelism in the α algorithm for process discovery by using the MPI programming model. Even though the proposed approach was able to achieve an average *speedup* of 3.94*x*, the upper limit of the *speedup* factor was limited by the number of independent steps available in the α algorithm.

An OpenMP application programming interface (API) based framework was proposed [54] to increase the execution efficiency of the α algorithm. The proposed framework not only exploited the task-parallelism but also the data-parallelism available in the α algorithm. With this modified approach the maximum *speedup* achieved was *speedup* and the upper limit of the *speedup* factor was limited by the number of unique events or activities discovered in an event log. However, the framework suffered to increase the *speedup* factor further due to the limited-bandwidth bottleneck induced by the in-memory computations available in the OpenMP model. The proposed OpenMP framework for the α algorithm was used over an artificially generated event log of size 2 GB. The details of the above mentioned three frameworks are listed in Table 3.

Table 3: Overview of different process discovery techniques in the absence of caseid.

S. No.	Method	Contributor(s)	Year	Tested on synthetic logs	Tested on real-world logs	In-memory computation	Applied on algorithm (s)	Event log size
1.	Map-Reduce	Evermann *et al.*, [19]	2014	Yes	No	No	α [62], FHM [76]	80 GB
2.	MPI framework	Sahu *et al.,* [53]	2018	Yes	No	Yes	α [62]	1.02 GB
3.	OpenMP framework	Sahu and Nayak [54]	2019	Yes	No	Yes	α [62]	2 GB

V. DISCUSSION

Based on the research questions Q1 and Q2 in Section II, we were able to find out sixty-one different process discovery techniques that relied on caseid and two other techniques that can discover process models in the absence of caseid respectively. Also, the search made for the research question Q3 yielded three proposals. Since our primary focus was Q2 and Q3, we have listed out our findings of the reviews on these two research questions below, which we believe, would be helpful for other researchers who are interested to work in the field of process mining. A probabilistic approach based on iterative Expectation-Maximization to tackle the challenge of discovering process models from unlabeled event logs presented [21].

However, the accuracy of their method relied on the following two factors: the total number of sources in the event log, and the number of overlapping sources. With a high number of sources, it is easier to discover consistent behavior in the event log. But, with anincreasing number of overlapping sources, it becomes difficult to separate the events belonging to different sources.

Two matrices namely the 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 not be able to correlate two activities [50].

The Map-Reduce framework for the α and the FHM algorithms proposed [19] is based on the distributed architecture and highly scalable. The proposed framework is more suitable and effective for processes where event log data are captured in a distributed fashion. However, for a centrally collected event log data, the proposed framework is not an efficient solution as it lacks in-memory computation.

The proposed MPI framework is a distributed architecture but allows in-memory computations. Thus, the framework is scalable as well as compatible with in-memory computations. But, the upper limit of the *speedup* factor achieved through this framework is limited by the number of independent tasks available in the α algorithm.

The OpenMP framework for the α algorithm produced a better execution efficiency in terms of the *speedup* factor as compared to the MPI framework [53, 54]. The OpenMP framework not only exploited the task-parallelism but also the data-parallelism available in the α algorithm. Specifically, this OpenMP based method targeted the number of unique activities and the causal relations available among those activities in an event log to exploit parallelism. Even though the framework was able to achieve a better *speedup*, the architecture itself is not highly scalable as the performance of this architecture degrades with increased memory traffic.

VI. CONCLUSION

In the last two decades, the field of process mining has attracted many researchers and a vast number of proposals have been published targeting many critical aspects of this topic. While the primary aim of the researchers was to develop process discovery methods, the secondary aim was to deal with critical issues such as detecting loops, splits-and-joins, and reducing noise. Similarly, the applications of process discovery algorithms have been widened as they are not confined to only the business process domain. Rather, they are being adopted in the fields like big-data, software engineering, and IoT.

VII. FUTURE SCOPE

The majority of the proposals in the field of process discovery rely on the caseid attribute of an event log to discover process models. Thus, the challenge of discovering process models in the absence of event logs is least explored and still open for future proposals. By discovering and gathering repeated event patterns from an unlabeled event log and establishing some kind of statistical correlations among those patterns may be considered to discover a process model.

Even though a few proposals are available which talk about increasing scalability and execution efficiency of the process discovery algorithms, the number of algorithms tested with these proposals is limited to the α and the FHM algorithm. Also, these proposals have their demerits. So, the future research directions in this regard would be: (i) to try to overcome the limitations of the existing proposals, (ii) to test the efficacy of the existing proposals on remaining process discovery algorithms, and (iii) to develop new execution frameworks.

ACKNOWLEDGMENTS

The authors would like to thank all the anonymous reviewers for their timely and constructive reviews.

Conflict of Interest. The authors do not have any conflict of interest with any other author.

REFERENCES

[1]. Abe, M., & Kudo, M. (2014). Business monitoring framework for process discovery with real-life logs. In *International Conference on Business Process Management*, 416-423.

[2]. Agrawal, R., Gunopulos, D., & Leymann, F. (1998, March). Mining process models from workflow logs. In *International Conference on Extending Database Technology*, 467-483.

[3]. De Medeiros, A. A., van Dongen, B. F., Van der Aalst, W. M., & Weijters, A. J. M. M. (2004). Process mining: Extending the α-algorithm to mine short loops.

[4]. Augusto, A., Conforti, R., Dumas, M., & La Rosa, M. (2017). Split miner: Discovering accurate and simple business process models from event logs. In *2017 IEEE International Conference on Data Mining (ICDM)*, 1-10.

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

[6]. Bernardi, M. L., Cimitile, M., Di Francescomarino, C., & Maggi, F. M. (2016). Do activity lifecycles affect the validity of a business rule in a business process?. *Information Systems*, *62*, 42-59.

[7]. Breuker, D., Matzner, M., Delfmann, P., & Becker, J. (2016). Comprehensible Predictive Models for Business Processes. *Mis Quarterly*, *40*(4), 1009-1034.

[8]. Buijs, J. C., van Dongen, B. F., & van der Aalst, W. M. (2013). Discovering and navigating a collection of process models using multiple quality dimensions. In *International Conference on Business Process Management,* 3-14, Springer, Cham.

[9]. Buijs, J. C., van Dongen, B. F., & van der Aalst, W. M. (2014). Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity. *International Journal of Cooperative Information Systems*, *23*(01), 1440001.

[10]. Carmona, J., & Cortadella, J. (2013). Process discovery algorithms using numerical abstract domains. *IEEE Transactions on Knowledge and Data Engineering*, *26*(12), 3064-3076.

[11]. Carmona, J., Cortadella, J., & Kishinevsky, M. (2009). New region-based algorithms for deriving bounded Petri nets. *IEEE Transactions on Computers*, *59*(3), 371-384.

[12]. Conforti, R., Dumas, M., García-Bañuelos, L., & La Rosa, M. (2016). BPMN Miner: Automated discovery of BPMN process models with hierarchical structure. *Information Systems*, *56*, 284-303.

[13]. Cook, J. E., & Wolf, A. L. (1998). Discovering models of software processes from event-based

data. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, *7*(3), 215-249.
[14]. Datta, A. (1998). Automating the discovery of as-is business process models: Probabilistic and algorithmic approaches. *Information Systems Research*, *9*(3), 275-301.
[15]. de Medeiros, A. K. A., Weijters, A. J., & van der Aalst, W. M. (2007). Genetic process mining: an experimental evaluation. *Data Mining and Knowledge Discovery*, *14*(2), 245-304.
[16]. De Smedt, J., De Weerdt, J., & Vanthienen, J. (2015). Fusion miner: Process discovery for mixed-paradigm models. *Decision Support Systems*, *77*, 123-136.
[17]. De Weerdt, J., De Backer, M., Vanthienen, J., & Baesens, B. (2012). A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs. *Information Systems*, *37*(7), 654-676.
[18]. Di Ciccio, C., & Mecella, M. (2013, April). A two-step fast algorithm for the automated discovery of declarative workflows. In *2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM),* 135-142.
[19]. Evermann, J. (2014). Scalable process discovery using map-reduce. *IEEE Transactions on Services Computing*, *9*(3), 469-481.
[20]. Ferilli, S. (2013). Woman: Logic-based workflow learning and management. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *44*(6), 744-756.
[21]. Ferreira, D. R., & Gillblad, D. (2009). Discovering process models from unlabelled event logs. In *International Conference on Business Process Management,* 143-158. Springer, Berlin, Heidelberg.
[22]. Ferreira, H. M., & Ferreira, D. R. (2006). An integrated life cycle for workflow management based on learning and planning. *International Journal of Cooperative Information Systems*, *15*(4), 485-505.
[23]. Folino, F., Greco, G., Guzzo, A., & Pontieri, L. (2009). Discovering expressive process models from noised log data. In *Proceedings of the 2009 international database engineering & applications symposium,* 162-172.
[24]. Gaaloul, W., Baïna, K., & Godart, C. (2005). Towards mining structural workflow patterns. In *International Conference on Database and Expert Systems Applications,* 24-33, Springer, Berlin, Heidelberg.
[25]. Goedertier, S., Martens, D., Vanthienen, J., & Baesens, B. (2009). Robust process discovery with artificial negative events. *Journal of Machine Learning Research*, *10*, 1305-1340.
[26]. Greco, G., Guzzo, A., Lupia, F., & Pontieri, L. (2015). Process discovery under precedence constraints. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *9*(4), 1-39.
[27]. Greco, G., Guzzo, A., & Pontieri, L. (2008). Mining taxonomies of process models. *Data & Knowledge Engineering*, *67*(1), 74-102.
[28]. Greco, G., Guzzo, A., & Pontieri, L. (2012). Process Discovery via Precedence Constraints. In *ECAI* (pp. 366-371).
[29]. Greco, G., Guzzo, A., Pontieri, L., & Sacca, D. (2006). Discovering expressive process models by clustering log traces. *IEEE Transactions on Knowledge and Data Engineering*, *18*(8), 1010-1027.
[30]. Günther, C. W., & Van Der Aalst, W. M. (2007). Fuzzy mining–adaptive process simplification based on multi-perspective metrics. In *International conference on business process management,* 328-343, Springer, Berlin, Heidelberg.
[31]. Guo, Q., Wen, L., Wang, J., Yan, Z., & Philip, S. Y. (2016). Mining invisible tasks in non-free-choice constructs. In *International Conference on Business Process Management,* 109-125, Springer, Cham.
[32]. Herbst, J., & Karagiannis, D. (2004). Workflow mining with InWoLvE. *Computers in Industry*, *53*(3), 245-264.
[33]. Huang, Z., & Kumar, A. (2012). A study of quality and accuracy trade-offs in process mining. *INFORMS Journal on Computing*, *24*(2), 311-327.
[34]. Hennessy, J. L., & Patterson, D. A. (2011). *Computer architecture: a quantitative approach*. Elsevier.
[35]. Lamma, E., Mello, P., Montali, M., Riguzzi, F., & Storari, S. (2007). Inducing declarative logic-based models from labeled traces. In *International Conference on Business Process Management,* 344-359, Springer, Berlin, Heidelberg.
[36]. Leemans, M., & van der Aalst, W. M. (2017). Modeling and discovering cancelation behavior. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems,* 93-113, Springer, Cham.
[37]. Leemans, S. J. J., Fahland, D., & Aalst, van der, W. M. P. (2014). Discovering block-structured process models from event logs containing infrequent behaviour. In N. Lohmann, M. Song, & P. Wohed (Eds.), *Business Process Management Workshops: BPM 2013 International Workshops, Beijing, China, August 26, 2013, Revised Papers*, 66-78. (Lecture Notes in Business Information Processing; Vol. *171*). Berlin: Springer.
[38]. Li, C., Ge, J., Huang, L., Hu, H., Wu, B., Yang, H., & Luo, B. (2016). Process mining with token carried data. *Information Sciences*, *328*, 558-576.
[39]. Liesaputra, V., Yongchareon, S., & Chaisiri, S. (2016). Efficient process model discovery using maximal pattern mining. In *International Conference on Business Process Management*, 441-456, Springer, Cham.
[40]. Maggi, F. M., Dumas, M., García-Bañuelos, L., & Montali, M. (2013). Discovering data-aware declarative process models from event logs. In *Business Process Management*, 81-96.
[41]. Maggi, F. M., Mooij, A. J., & van der Aalst, W. M. (2011). User-guided discovery of declarative process models. In *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, 192-199.
[42]. Maggi, F. M., Slaats, T., & Reijers, H. A. (2014). The automated discovery of hybrid processes. In *International Conference on Business Process Management*, 392-399.
[43]. Mannhardt, F., de Leoni, M., Reijers, H. A., & 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, 545-560.
[44]. Mannila, H., & Meek, C. (2000). Global partial orders from sequential data. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining,* 161-168.
[45]. Măruşter, L., Weijters, A. T., Van Der Aalst, W. M., & Van Den Bosch, A. (2006). A rule-based approach for process discovery: Dealing with noise and imbalance in process logs. *Data mining and knowledge discovery*, *13*(1), 67-87.
[46]. Mokhov, A., Carmona, J., & Beaumont, J. (2016). Mining conditional partial order graphs from event logs. In *Transactions on Petri Nets and Other Models of Concurrency 11*, 114-136.
[47]. Molka, T., Redlich, D., Drobek, M., Zeng, X. J., & Gilani, W. (2015). Diversity guided evolutionary mining of hierarchical process models. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, 1247-1254.
[48]. Nguyen, H., Dumas, M., ter Hofstede, A. H., La Rosa, M., & Maggi, F. M. (2017, June). Mining business process stages from event logs. In *International Conference on Advanced Information Systems Engineering*, 577-594.
[49]. Pesic, M., Schonenberg, H., & Van der Aalst, W. M. (2007). Declare: Full support for loosely-structured processes. In *11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007)*, 287-287.
[50]. Pourmirza, S., Dijkman, R., & Grefen, P. (2017). Correlation miner: mining business process models and event correlations without case identifiers. *International Journal of Cooperative Information Systems*, *26*(02), 1742002.
[51]. Redlich, D., Molka, T., Gilani, W., Blair, G. S., & Rashid, A. (2014). Scalable Dynamic Business Process Discovery with the Constructs Competition Miner. In *SIMPDA*, 91-107.
[52]. Rubin, V., Van Dongen, B. F., Kindler, E., & Günther, C. W. (2006). Process mining: A two-step approach using transition systems and regions. In *BPM Center Report BPM-06-30, BPM Center*.
[53]. Sahu, M., Chakraborty, R., & Nayak, G. (2018). A task-level parallelism approach for process discovery. *International Journal of Engineering & Technology*, *7*(4), 2446-52.
[54]. Sahu, M., & Nayak, G. K. (2019). An efficient parallel framework for process discovery using OpenMP. *International Journal of Advanced Computer Research*, *9*(41), 112-123.
[55]. Santhoshkumar, S., Mohamed, A. T., & Ramaraj, E. (2019). Process Analytics Model for Health Care using IoT and Big Data Techniques. *International Journal on Emerging Technologies*, *10*(4), 197–200.
[56]. Schimm, G. (2002). Process miner—a tool for mining process schemes from event-based data. In *European Workshop on Logics in Artificial Intelligence*, 525-528.
[57]. Schimm, G. (2004). Mining exact models of concurrent workflows. *Computers in Industry*, *53*(3), 265-281.
[58]. Schönig, S., Rogge-Solti, A., Cabanillas, C., Jablonski, S., & Mendling, J. (2016). Efficient and customisable declarative process mining with SQL. In *International Conference on Advanced Information Systems Engineering,* 290-305.
[59]. Song, W., Jacobsen, H. A., Ye, C., & Ma, X. (2015). Process discovery from dependence-complete event logs. *IEEE Transactions on Services Computing*, *9*(5), 714-727.
[60]. Tapia-Flores, T., Rodríguez-Pérez, E., & López-Mellado, E. (2016). Discovering Process Models from Incomplete Event Logs using Conjoint Occurrence Classes. In *ATAED@ petri nets/ACSD*, 31-46.
[61]. Van Der Aalst, W. (2016). Data science in action. In *Process mining*, 3-23.
[62]. Van der Aalst, W., Weijters, T., & Maruster, L. (2004). Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, *16*(9), 1128-1142.
[63]. Van der Aalst, W. M., Rubin, V., Verbeek, H. M. W., van Dongen, B. F., Kindler, E., & Günther, C. W. (2010). Process mining: a two-step approach to balance between underfitting and overfitting. *Software & Systems Modeling*, *9*(1), 87.
[64]. Van der Aalst, W. M., & Weijters, A. J. (2004). Process mining: a research agenda.
[65]. Werf, van der, J. M. E. M., Dongen, van, B. F., Hurkens, C. A. J., & Serebrenik, A. (2009). Process discovery using integer linear programming. *FundamentaInformaticae*, *94*(3-4), 387-412.
[66]. Van Dongen, B. BPI challenge 2012 dataset (2012).
[67]. van Dongen, B. F., & Van der Aalst, W. M. (2005). Multi-phase process mining: Aggregating instance graphs into EPCs and Petri nets. In *PNCWB 2005 workshop*, 35-58.
[68]. van Eck, M. L., Buijs, J. C., & van Dongen, B. F. (2014). Genetic process mining: Alignment-based process model mutation. *International Conference on Business Process Management*, 291-303.
[69]. van Eck, M. L., Sidorova, N., & van der Aalst, W. M. (2016). Discovering and exploring state-based models for multi-perspective processes. In *International Conference on Business Process Management,* 142-157.
[70]. van Zelst, S. J., van Dongen, B. F., van der Aalst, W. M., & Verbeek, H. M. W. (2018). Discovering workflow nets using integer linear programming. *Computing*, *100*(5), 529-556.
[71]. vandenBroucke, S. K., & De Weerdt, J. (2017). Fodina: A robust and flexible heuristic process discovery technique. *decision support systems*, *100*, 109-118.
[72]. vandenBroucke, S. K., Vanthienen, J., & Baesens, B. (2014). Declarative process discovery with evolutionary computing. In *2014 IEEE Congress on Evolutionary Computation (CEC)* (pp. 2412-2419). IEEE.
[73]. Vasilecas, O., Savickas, T., & Lebedys, E. (2014). Directed acyclic graph extraction from event logs. In *International Conference on Information and Software Technologies,* 172-181.
[74]. Vázquez-Barreiros, B., Mucientes, M., & Lama, M. (2015). ProDiGen: Mining complete, precise and minimal structure process models with a genetic algorithm. *Information Sciences*, *294*, 315-333.
[75]. Verbeek, H. M. W., van der Aalst, W. M., & Munoz-

Gama, J. (2017). Divide and conquer: A tool framework for supporting decomposed discovery in process mining. *The Computer Journal*, *60*(11), 1649-1674.
[76]. Weijters, A. J. M. M., & Ribeiro, J. T. S. (2011). Flexible heuristics miner (FHM). In *2011 IEEE symposium on computational intelligence and data mining (CIDM)*, 310-317.
[77]. Weijters, A. J. M. M., van Der Aalst, W. M., & De Medeiros, A. A. (2006). Process mining with the heuristics miner-algorithm. *Technische Universiteit Eindhoven, Tech. Rep. WP*, *166*, 1-34.
[78]. Wen, L., Van Der Aalst, W. M., Wang, J., & Sun, J. (2007). Mining process models with non-free-choice constructs. *Data Mining and Knowledge Discovery*, *15*(2), 145-180.
[79]. Wen, L., Wang, J., van der Aalst, W. M., Huang, B., & Sun, J. (2009). A novel approach for process mining based on event types. *Journal of Intelligent Information Systems*, *32*(2), 163-190.
[80]. Westergaard, M., & Maggi, F. M. (2011). Declare: A Tool Suite for Declarative Workflow Modeling and Enactment. *BPM (Demos)*, 1-5.
[81]. Yahya, B. N., Bae, H., Sul, S. O., & Wu, J. Z. (2013). Process discovery by synthesizing activity proximity and user's domain knowledge. In *Asia-Pacific Conference on Business Process Management* , 92-105.
[82]. Yahya, B. N., Song, M., Bae, H., Sul, S. O., & Wu, J. Z. (2016). Domain-driven actionable process model discovery. *Computers & Industrial Engineering, 99*, 382–400.

How to cite this article: 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.