PROCESS MINING ALGORITHMS

Asst. Prof. Esmita Gupta

M.E. Student, Vidyalankar Institute of Technology, Mumbai University, (India).

ABSTRACT

The systems that support today's globally distributed and agile businesses are steadily growing in size and generating numerous events. Business Intelligence aims to support and improve decision making processes by providing methods and tools for analyzing the data. Process mining is a new research technology that sits between computational intelligence and data mining on the one hand, and process modelling and analysis on the other hand. Its primary objective is the discovery of process models based on available event log data.

Various process mining algorithms have been proposed recently, but there does not exist a widely-accepted benchmark to evaluate and compare these process mining algorithms. As a result, it can be difficult to choose a suitable process mining algorithm for a given enterprise or application domain.

This paper proposes a solution to evaluate and compare these process mining algorithms efficiently, so that businesses can select the process mining algorithms that are most suitable for a given model set to take good decisions for their organization.

Keywords: Process Mining; Heuristic Miner; Genetic Miner; Fuzzy Miner

I. INTRODUCTION

Companies use information systems to enhance the processing of their business transactions. Business processes cannot be evaluated without the integration of information systems that support and monitor relevant activities in modern companies. The increasing integration of information systems does not only provide the means to increase effectiveness and efficiency. It also opens up new possibilities of data access and analysis. When information systems are used for supporting and automating the processing of business transactions they generate data. These data need to be evaluated and processed in a proper manner in order to make good decision for the growth of organization.

The idea of process mining is to discover, monitor and improve real processes by extracting knowledge from the extracted data and generate event logs. The main benefit of process mining techniques is that information is objectively compiled. In other words, process mining techniques are helpful because they gather information about what is actually happening according to an event log of an organization, and not what people think that is happening in the organization.

Process mining algorithm helps in processing the required data and different types of models can be extracted that describe underlying processes. A business process model shows a specific ordering of work activities with a beginning, an end, and clearly defined inputs and outputs.

Here, we provide a short introduction to process mining, including the working of three established process-mining algorithms. Different process mining algorithms have different properties (e.g., some perform better on models with invisible tasks, while others do not), being able to select the most appropriate process mining

algorithm (i.e. the one that produces mined models that are semantically similar to the original models and structurally equal to or better than the original models) for the models from a given enterprise is a big advantage [5].

The organization of the rest of this paper is as follows. Section 2 discusses why process mining. Section 3 discusses about the working of heuristic, genetic and fuzzy miner algorithm each with an illustration in order to understand the concept.

II WHY PROCESS MINING

The emergence of semi-structured processes, combined with improvements in computing power and the speed of data transmission, have fueled the need for mining algorithms to address new challenges.

In addition to discovering process models from logs and examining the level of conformance of an actual process to its modeled counterpart, process mining should find, merge, and clean event data; handle changes in a process that occur while its being mined; and provide operational support to process users in an online manner.[1]

1. The first challenge is that numerous uncorrelated events (with possible noise) are gathered from disparate heterogeneous
   and distributed systems.
2. The second challenge is the Dramatic variation in execution behavior.
3. The third challenge is parallel and repeated task execution.

Thus this topic includes a method for systematic comparison, and a body of experimental data describing the behavior of three algorithms i.e. Heuristic Miner, Fuzzy miner and Genetic Miner with typical process structures and how it can handle the above challenges.

The primary aim of process mining is to investigate a scalable solution that can evaluate, compare and rank these process mining algorithms efficiently. In particular, it attempts to investigate how we can choose an effective process mining algorithm for an enterprise without evaluating different process mining algorithms.

III PROCESS MINING ALGORITHM

There are many different approaches to process mining. Local methods look at local relations between activities in logs(Heuristics Miner), while global approaches build and refine a model based on the whole log (Genetic Mining, Fuzzy Miner) Different algorithms have their own specialism. Heuristics Miner uses frequencies and parameterization to handle noise; while Genetic Process Mining can mine complex and noisy logs, but is resource intensive. More recent approaches focus on managing complex real world models or noisy logs using clustering and abstraction, e.g. at the trace or activity level.

In this paper we are going to objectively differentiate the most used algorithms with examples.

1. Heuristic miner
2. Genetic miner
3. Fuzzy miner

3.1. Heuristic Miner

Heuristics Miner is a practical applicable mining algorithm that can deal with noise, and can be used to express the main behavior that is not all details and exceptions, registered in an event log. This technique extends alpha algorithm by considering the frequency of traces in the log. The Heuristics Miner Plug-in mines the control flow perspective of a process model. To do so, it only considers the order of the events within a case. For instance for the log in the log file only the fields case id, time stamp and activity are considered during the mining. The timestamp of an activity is used to calculate these orderings [9].

Fig 1. Working of Heuristic Miner algorithm

It includes three steps:

(1) The construction of the dependency graph,

(2) For each activity, the construction of the input and output expressions and

(3) The search for long distance dependency relations [6].

This is particularly useful in the context of scientific workflows, just as the running example, many scientific workflows have multiple tasks even hundreds of tasks scheduled in parallel, not each parallel task succeed the dependent task directly in provenance, therefore, long distance dependency discovery is especially important in the context of scientific workflows.

The Heuristics Miner Plug-in mines the control-flow perspective of a process model. To do so, it only considers the order of the events within a case. For instance it creates an event log table with the fields as case id, originator of the activity, time stamp and activities that are considered during the mining. The timestamp of an activity is used to calculate these ordering.

Steps of Heuristic Miner algorithm

1. Read a log
2. Get the set of tasks
3. Infer the ordering relations based on their frequencies
4. Build the net based on inferred relations
5. Output the net

As we know, Process mining techniques are able to extract knowledge from event logs. To find a process model on the basis of an event log, the log should be analysed for causal dependencies, e.g., if an activity is always followed by another activity it is likely that there is a dependency relation between both activities. To analyse these relations we introduce the following notations. Let T be a set of activities. S ϵ T is an event trace, i.e., an

arbitrary sequence of activity identifiers. W Ɛ T is an event log, i.e., a multiset that is bag of event traces. Note that since W is a multiset, every event trace can appear more than once in a log [9].Lets analyse these relations by introducing the following notations. Let W be an event log over T, i.e., W Ɛ T. Let a,b ƐT

1. a >W b if and only if there is a trace S = t1,t2,t3 : : : tn and i Ɛ {1; : : : ; n -1} such that S ƐW and ti = a and ti+1 = b,
2. a →W b if and only if a >W b and b > not equal W a,
3. a ≠ W b if and only if a > not equal W b and b > not equal W a, and
4. a ‖ W b if and only if a >W b and b >W a.
5. a>>W b if and only if there is a trace S = t1,t2,t3 ….. tn And i Ɛ {1; : : : ; n - 2} such that S ϵ W andti = a and ti+1 = b and

   ti + 2 = a
6. a >>>W b if and only if there is a trace S = t1, t2, t3….tn and i < j and i, j ϵ {1……n} such that S Ɛ W and ti = a and tj = b.

Basic ordering relations:

- Direct succession: x>y iff for some case x is directly followed by y.
- Causality: x→y iff x>y and not y>x.
- Parallel: x||y iff x>y and y>x
- Unrelated: x#y iff not x>y and not y>x.

After inferring the relation according to the frequencies we start with the construction of a so called dependency graph. A frequency based metric is used to indicate how certain we are that there is truly a dependency relation between two events ‘a’ and ‘b’ (notation a$\Rightarrow$Wb).The calculated $\Rightarrow$W values between the events of an event log are used in a heuristic search for the correct dependency relations.

1. Let W be an event log over T, and a, b ϵ T. Then |a >W b |is the number of times a >W b occurs in W, and,

$$a \Rightarrow w\, a = \left(\frac{|a>wb|-|b>wa|}{|a>wb|+|b>wa|+1}\right) \quad (1)$$

Where a$\Rightarrow$Wb is always between 1 and -1.

2. For short loops, the dependency is measured as follows, Let W be an event log over T, and a, b ϵ T Then |a >W a| is the number of times a >W a occurs in W, and |a >>W a| is the number of times a >>W b occurs in W.

$$a \Rightarrow w\, a = \left(\frac{|a>wa|}{|a>wa|+1}\right) \quad (2)$$

$$a \Rightarrow_{2W} b = \left(\frac{|a\gg wb|+|b\gg wa|}{|a\gg wb|+|b\gg wa|+1}\right) \quad (3)$$

The event log W is a bag, the same trace can appear more than once in the log and patterns can appears more times in a trace. If, for instance, the pattern ‘ab’ appears twice in a trace (e.g., cabefgcabefh), and this trace appears three times in W (i.e. W (cabefgcabefh) =3) then these appearances count as 6 in the a >W b measurement. After calculating the dependency value for the given log, the dependency table is as follows:

1. The given log is:

$L=[(a,e)^{5}, (a,b,c,e)^{10}, (a,c,b,e)^{10}, (a,b,e)^{1}, (a,c,e)^{1}, (a,d,e)^{10}, (a,d,d,e)^{2}, (a,d,d,d,e)^{1}]$

Table 1: Log table with frequencies

|>L|	a	b	c	d
a	0	11	11	13
b	0	0	10	0
c	0	10	0	4
d	0	0	0	0
e	0	0	0	0

Table 2: Dependency relation matrix

|>L|	a	b	c	d	e
a	0	0.92	0.92	0.93	0.83
b	-0.92	0	10	0	0.92
c	-0.92	0	0	0	0.92
d	-0.93	0	0	0.80	0.93
e	-0.83	-0.92	-0.92	-0.93	0

Thus the heuristic graph is

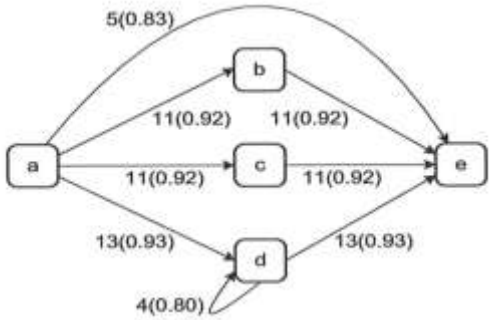

Fig 2: Heuristic graph [12]

The process mining is the process of generating new process model for the benefit of business processing system. Heuristics Miner is used to generate a new model without noise that is with exception, frequent failure or partial failure, life time of the model, etc. Hence, Heuristics Miner is a most practical applicable mining algorithm that can deal with noise, and can be used to express the main behaviour that is not all details and exceptions, registered in an event logs.

3.2. Genetic Miner

The process mining has attracted the attention of both researchers and tool vendors in the Business Process Management (BPM) space. The goal of process mining is to discover process models from event logs, i.e., events logged by some information system are used to extract information about activities and their causal relations. But these models were not able to tackle with noise, incompleteness, duplicate activities, hidden activities, non-free-choice constructs, etc. To tackle these problems we propose a completely new approach based on genetic algorithms. As can be expected, a genetic approach is able to deal with noise and incompleteness.

Genetic miner uses a genetic algorithm to mine a petri net representation of the process model from execution traces. Genetic mining algorithms use an evolutionary approach that mimics the process of natural evolution [2]. A genetic search is an example of a global search strategy because the quality or fitness of a candidate model is calculated by comparing the process model with all traces in the event log the search process takes place at a global level. For a local strategy there is no guarantee that the outcome of the locally optimal steps (at the level

of binary event relations) will result in a globally optimal process model. Hence, the performance of such local mining techniques can be seriously hampered when the necessary information is not locally available because one erroneous example can completely mess up the derivation of a right model. Therefore, we started to use genetic algorithms.

Genetic algorithm is used to discover a Petri net given a set of event traces. Genetic algorithms are adaptive search methods that try to mimic the process of evolution [9].

These algorithms start with an initial population of individuals (in this case process models). Populations evolve by selecting the fittest individuals and generating new individuals using genetic operators such as crossover (combining parts of two of more individuals) and mutation (random modification of an individual) [8].

The goal of using genetic algorithms is to tackle problems such as duplicate activities, hidden activities, non-free-choice constructs, noise, and incompleteness, i.e., overcome the problems of some of the traditional approaches.

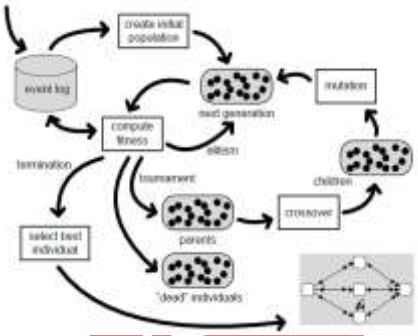

Fig 3. Genetic Process Mining [12]

3.2.1Genetic Miner Algorithm:

When using genetic algorithms to mine process models, there are three main concerns.

The first is to define the internal representation. The internal representation defines the search space of a genetic algorithm.

The second concern is to define the fitness measure. In our case, the fitness measure evaluates the quality of a point (individual or process model) in the search space against the event log.

The third concern relates to the genetic operators (crossover and mutation) because they should ensure that all points in the search space defined by the internal representation may be reached when the genetic algorithm runs.

1. Read event log
2. Build the initial population
3. Calculate fitness of the individuals in the population
4. Stop and return the fittest individuals
5. Create next population – use elitism and genetic operators

Let's consider an event log as follows:

Table 3: Event log for Genetic process mining

CaseId	ActivityId	Originator	Timestamp	CaseId	ActivityId	Originator	Timestamp
1	A	John	09-03-04	3	E	Peter	10-03-04
2	A	John	09-03-04	3	F	Carol	11-03-04
3	A	Sue	09-03-04	4	D	Peter	11-03-04
3	D	Carol	09-0304	3	G	Sue	11-03-04
1	B	Mike	09-03-04	3	H	Peter	11-03-04
1	H	John	10-03-04	4	F	Sue	11-03-04
2	C	Mike	10-03-04	4	E	Clare	11-03-04
4	A	Sue	10-03-04	4	G	Mike	11-03-04
2	H	John	10-03-04	4	H	Clare	11-03-04

The above Event log is present with the timestamp factor in order to know the person who carried out a task at what date and time. The corresponding Petri net graph with work flow model that is generated from the above given event log is:

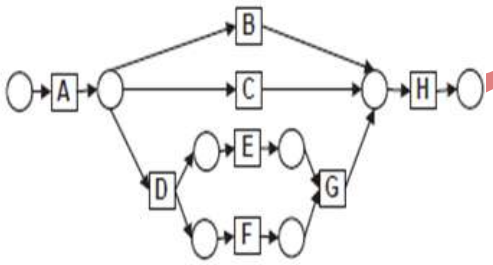

Fig 4. Petri Net Graph of Table3

Step 1: Initial Population

The initial population is randomly built by the genetic algorithm. Individuals are causal matrices. While building the initial population we follow the following definition:

A Causal Matrix is a tuple CM = (A, C, I, O), where

- A is a finite set of activities,
- $C \subseteq A \times A$ is the causality relation,
- $I \in A \rightarrow P(P(A))$ is the input condition function,
- $O \in A \rightarrow P(P(A))$ is the output condition function

Given a log, all individuals in any population of the genetic algorithm have the same set of activities (or tasks) A. This set contains the tasks that appear in the log. However, the causality relation C and the condition functions I and O may be different for every individual in the population. The initial population can have any individual in the search space defined by a set of activities A. The higher the amount of tasks that a log contains, the bigger is the search space.

Given the event log in Table 3, Table 4 shows two individuals that could be created during the initialization.

Table 4: Two randomly created individuals for the log in Table 3

Individual 1			Individual 2		
Activity	Input	Output	Activity	Input	Output
A	{}	{{B,C,D}}	A	{}	{{B,C,D}}
B	{{A}}	{{H}}	B	{{A}}	{{H}}
C	{{A}}	{{H}}	C	{{A}}	{{H}}
D	{{A}}	{{E}}	D	{{A}}	{{E,F}}
E	{{D}}	{{G}}	E	{{D}}	{{G}}
F	{}	{{G}}	F	{{D}}	{{G}}
G	{{E},{F}}	{{H}}	G	{{E},{F}}	{{H}}
H	{{C,B,C}}	{}	H	{{C},{B},{C}}	{}

Step 2: Fitness calculation

If an individual in the genetic population correctly describes the registered behavior in the event log, the fitness of that individual will be high. In our approach the fitness is strongly related to the number of correctly parsed traces from the event log.

We make use of parsing semantics because it is more robust to noisy logs and it gives more information about the fitness of the complete process models (i.e not biased to only the first part of the process model). In a noise-free situation, the fitness of a model can be 1 (or 100%) (i.e. all traces can be parsed). In practical situations, the fitness value ranges from 0 to 1. The exact fitness of an individual to a given log is given by the formula:

$$\text{Fitness} = 0.40 * \frac{\text{allParsedActivities}}{\text{NumberOfActivitiesAtLog}} + 0.60 * \frac{\text{allProperlyCompletedLog Traces}}{\text{NumberOfTracesAtLog}}$$

where: numberOfActivitiesAtLogin the number of activities in the log. For instance, the log shown in Table 4 has 18 activities. numberOfTracesAtLogin the number of log traces, e.g., in Table 4 there are 4. allParsedActivities is the sum of parsed activities (i.e. activities that could fire without the artificial addition of tokens) for all log traces. allProperlyCompletedLogTraces is the number of log traces that were properly parsed (i.e. after the parsing the end place is the only one to be marked).

Step 3: Genetic operation

We use elitism, crossover and mutation to build the population elements of the next genetic generation.

- **Elitism** means that a percentage of the fittest individuals in the current generation is copied into the next generation.
- **Crossover** and mutation are the basic genetic operators. Crossover creates new individuals (offsprings) based on the fittest individuals (parents) in the current population. So, crossover recombines the fittest material in the current population in the hope that the recombination of useful material in one of the parents will generate an even fitter population element. The mutation operation will change some minor details of a population element. The hope is that the mutation operator will insert new useful material in the population.

- The genetic algorithm (GA) stops when:

(i) It finds an individual whose fitness is 1; or

(ii) It computes n generations, where n is the maximum number of generation that is allowed; or

(iii) The fittest individual has not changed for n/2 generations in a row.

If none of these conditions hold, the GA creates a new population as follows:

Input: current population, elitism rate, crossover rate and mutation rate

Output: new population

1. Copy "elitism rate × population size" of the best individuals in the current population to the next population.

2. While there are individuals to be created do:

(a) Use tournament selection to select parent1.

(b) Use tournament selection to select parent2.

(c) Select a random number r between 0 (inclusive) and 1 (exclusive).

(d) If r is less than the crossover rate:

Then do crossover with parent1 and parent2. This operation generates two offsprings: offspring1 and offspring2.else offspring1 equals parent1 and offspring2 equals parent2.

(e) Mutate offspring1 and offspring2. (This step is only needed if the mutation rate is non-zero.)

(f) Copy offspring1 and offspring2 to the new population.

3. Return the new population.

Tournament Selection To select a parent the tournament selection algorithm randomly selects 5 individuals and returns the fittest individual among the five ones.

Crossover The most important and complex genetic operation in our genetic approach is the crossover operation. Starting point of the crossover operation are the two parents (i.e. parent1 and parent2). The result of applying the crossover operation is two offsprings (offspring1 and offspring2).

The **mutation** works on the INPUT and OUTPUT boolean expressions of an activity. For every activity t in an individual, a new random number r is selected. Whenever r is less than the "mutation rate" 7, the subsets in INPUT (t) are randomly merged or split. The same happens to OUTPUT (t). As an example, consider Offspring1 in Table 5. Assume that the random number r was less than the mutation rate for activity D. After applying the mutation, OUTPUT (D) changes from {{E, F}} to {{E}, {F}}. Note that this mutation does not change an individual's causal relations, only its AND-OR/join-split may change.

3.3. Fuzzy miner

Process Mining is a technique for extracting process models from execution logs. This is particularly useful in situations where people have an idealized view of reality. Real-life processes turn out to be less structured than people tend to believe. Unfortunately, traditional process mining approaches have problems dealing with unstructured processes. The discovered models are often "spaghetti-like", showing all details without distinguishing what is important and what is not. This topic proposes a new process mining approach to overcome this problem. The approach is configurable and allows for different faithfully simplified views of a particular process. To do this, the concept of a roadmap is used as a metaphor [10].

Fuzzy Miner is suitable for mining less structured processes which exhibit a large amount of unstructured and conflicting behaviour i.e. spaghetti-like models into more concise models. It applies a variety of techniques, such as removing unimportant edges, clustering highly correlated nodes in to a single node, and removing isolated node clusters. [2]

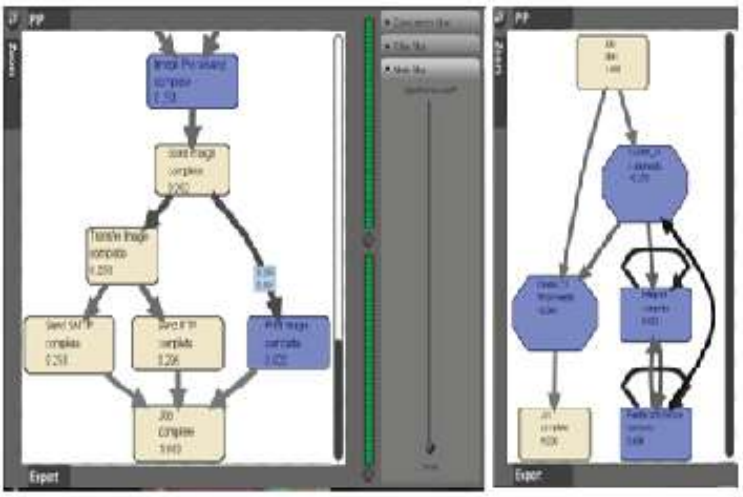

Fig 5. Fuzzy model and Fuzzy instance [7]

An Adaptive Approach for Process Simplification:

- Process mining techniques which are suitable for less-structured environments need to be able to provide a high-level view on the process, abstracting from undesired details.
- Activities in a process can be related to locations in a topology (e.g. towns or road crossings) and
- Precedence relations to traffic connections between them (e.g., railways or motorways).

When one takes a closer look at maps, the solution cartography has come up with to simplify and present complex topologies; one can derive a number of valuable concepts from them.

- Aggregation: To limit the number of information items displayed, maps often show coherent clusters of low-level detail information in an aggregated manner. One example are cities in road maps, where particular houses and streets are combined within the city's transitive closure.
- Abstraction: Lower-level information which is insignificant in the chosen context is simply omitted from the visualization. Examples are bicycle paths, which are of no interest in a motorist's map.
- Emphasis: More significant information is highlighted by visual means such as colour, contrast, saturation, and size. For example, maps emphasize more important roads by displaying them as thicker, more colorful and contrasting lines.
- Customization: There is no one single map for the world. Maps are specialized on a defined local context, have a specific level of detail (city maps vs. highway maps), and a dedicated purpose.

3.3.1 Fuzzy miner algorithm:

Input

A set of N transactions, each with n attribute, fuzzy linguistic terms for quantitative attributes, The user-specified minimum fuzzy support, The user-specified minimum fuzzy confidence, A domain Ontology.

Output

Phase I: Fuse similar behaving attributes;

Phase II: Generate Meta rules;

Phase III: Generate frequent fuzzy itemsets;

Phase IV: Make fuzzy association rules.

- The Fuzzy Miner is a process miner that mines an event log for a family of process models using a “map" metaphor.
- Let’s assume a single map as fuzzy instance whereas we call a family of maps as fuzzy model.
- Like high-level maps only show major objects of interest and major streets, high-level fuzzy instances also show only major elements (nodes and edges).
- For this purpose, the Fuzzy Miner computes from the log a significance weight for every element and an additional correlation weight for every edge.
- The higher these weights are, the more major the element is considered to be.
- Furthermore, the Fuzzy Miner uses a number of thresholds: Only elements that meet these thresholds are shown.
- Fuzzy Miner first attempts to cluster minor nodes into major cluster nodes, and only if that does not work it will remove the minor node from the map.

The Fuzzy Miner has been enhanced to utilize the availability of sub-logs obtained from the Pattern Abstractions plugin for the chosen abstractions. Fuzzy models are discovered for each of the sub-logs and are displayed upon zooming in on its corresponding abstract activity. Abstract activities are differentiated from other activities by means of a distinct color.

Thus we can say that fuzzy miner successfully gives the changed part and unchanged part, giving the most dependency correctly.

IV CONCLUSION AND FUTURE WORK

The significance of process mining increases with the growing integration of information systems.

Many process mining techniques have been proposed. It is difficult to know which algorithms are better.

Thus here we have carried out a comparative study to identify which algorithm to be used and when. After the study we can say that

- Heuristic miner can be used when we have real-life data with not too many different events or when you need a Petri net model for further analysis
- Genetic Miner can be used when we need to mine logs with noise, handling of duplicate task names, local and nonlocal nonfree choice constructs and invisible task.
- Fuzzy miner can be used when we have seen various algorithms (i) depict desired traits, (ii) eliminate irrelevant details, (iii) reduce complexity, and (iv) improve comprehensibility.

Process mining is still a young research discipline. Although a model mined by a process-mining algorithm can present a picture of a process’s execution at a particular point in time, as the process changes and evolves, the mined model might no longer maintain fidelity to the process. This leads to the challenge of handling changes in the process (also known as concept drift) as the process is being analyzed. The future scope of our topic will be

to have runtime changes in the process model with the help of these algorithms with monitoring, verification of requirements, and compliance enforcement.

REFERENCES

[1] IEEE Taskforce "Process mining manifesto", 2012

[2] Geetika T. Lakshmanan and Rania Khalaf, "Leveraging Process mining techniques", IBM T.J. Watson Research Center , IT Pro September/October 2013

[3] StB Prof. Dr. Nick Gehrke, Michael Werner, Dipl.-Wirt.-Inf. "Process Mining", *WISU - die Zeitschrift für den Wirtschaftsstudenten 7/13*

[4] Philip Weber, BehzadBordbar, Peter Tino, BasimMajeed "A Framework for comparing Process Mining Algorithm"

[5] Jianmin Wang, Raymond K. Wong, Jianwei Ding, QinlongGuo and Lijie Wen "Efficient selection of mining algorithm", IEEE Transactions on Services Computing 01/2013

[6] RengZeng, Xudong He, Jiafei Li, Zheng Liu, W.M.P. van der Aalst "A Method to Build and Analyze Scientific Workflows from Provenance through Process Mining", Proceedings of the 3rd USENIX Workshop on the Theory and Practice of Provenance (TaPP'11); 01/2011

[7] R.P. Jagadeesh Chandra Bose, Eric H.M.W. Verbeek1 and Wil M.P. van der Aalst1 "Discovering Hierarchical Process Models Using ProM", Department of Mathematics and Computer Science, University of Technology, Eindhoven, The Netherlands, Conference: Proceedings of the CAiSE Forum 2011.

[8] W.M.P. van der Aalst, A.K. Alves de Medeiros, and A.J.M.M. Weijters "Genetic Process Mining", Department of Technology Management, Eindhoven University of Technology, 26th International Conference, ICATPN 2005

[9] Saravanan .M.S, Rama Sree .R.J "A Role of Heuristics Miner Algorithm in the Business Process System", Comp. Tech. Appl., Vol 2 (2), 340-344

[10] W.M.P. van der Aalst*, A.J.M.M. Weijters "Process mining: a research agenda", Department of Technology Management, Eindhoven University of Technology,2004.

[11] Fabrizio Maria Maggi "Process Mining-Control flow process discovery", Springer 2011

[12] Ana Karla Alves de Medeiros,"Process Mining-Control flow mining algorithm", Eindhoven University of Technology

[13] Christian W. Gunther and Wil M.P. van der Aalst "Fuzzy Mining – Adaptive Process Simplification Based on Multi-Perspective Metrics", Eindhoven University of Technology.

[14] Dr. Anne Rozinat and Dr. Christian W. Günther "The Added Value of Process Mining", 2014.

[15] "Introduction to Process Mining, turning big data in real value.mp4", www.processmining.org, 04 Aug 2014.

[16] "Fuzzy miner", www.fluxicon.com, 13 Sept 2014.

[17] "Which mining algorithm should you use", www.processmining.org, 04 Aug 2014.

[18] "How to get started with process mining", www.processmining.org, 04 Aug 2014.