Process Profiling based Synthetic Event Log Generation

Eren Esgin[1,2] and Pinar Karagoz[3]
[1]MBIS R&D Center, AI Research, Istanbul, Turkey
[2]Middle East Technical University, Informatics Institute, Ankara, Turkey
[3]Middle East Technical University, Computer Engineering Department, Ankara, Turkey

Keywords: Case Identifier, Process Mining, Process Profile, Synthetic Log Generator (SynLogGen), Unlabeled Event Log.

Abstract: The goal of process mining is to discover the process behavior from the runtime information of process executions. Having labeled process logs is crucial for process mining research. However, real life event logs at process-aware information systems are mostly partially assigned to case identifiers, known as unlabeled event log problem. As a remedy to labeled data need in process mining research, we propose an approach to generate synthetic event logs according to the provided process profile, which outlines the activity vocabulary and structure of the corresponding business process. We evaluate the performance of our prototypical implementation in term of compatible log generation under varying parameter setting complexities.

1 INTRODUCTION

The impact of process-aware information systems is limited by the difficulties encountered at process design phase (Weijters & van der Aalst, 2003). Respectively, reference process models are normative in the sense that they reflect what should be done rather than the actual process execution (van der Aalst et al., 2003). Instead of manually designing the corresponding process, it is proposed to reverse this traditional design procedure by a more objective and automated technique called *process mining*. Process mining collects the process knowledge and distills the process patterns from the low-level process history. The major assumption about process mining is the existence of *event log*, which contains a sequence of process trails in the form <*caseID, taskID*> where caseID identifies the process instance and taskID specifies the activity that has been performed (Walicki & Ferreira, 2011). A case identifier is important to correlate different events recorded in the event log and this orchestration of process instances is called *labeled event log* (Bayomie et al., 2016).

In capability maturity levels at software development process, providing event logs with automatically assigned case identifiers is classified as maturity level-4 or higher at logging (van der Aalst, 2011; Dustdar & Gombortz, 2006). However, the process may execute in an environment of lower logging maturity level. As stated in (van der Aalst, 2006), the event logs of most enterprise resource planning (ERP) vendors are unable to monitor unique and individual process cycles. Instead, they only log the execution of transactions without referring the corresponding case. This is due to the fact that, these systems are mostly *data centric* such that the event logs are staggered at the application tables with the lack of case relations. Such kind of logs is called as *unlabeled event log* (van der Aalst et al., 2006; Gunther & van der Aalst, 2006). Limitations in obtaining labeled data constitutes an important drawback for process mining research, especially for supervised learning-based approaches, which call for labeled training data set.

In this paper, we address the challenge of extracting and preparing event logs for analysis. Rather than correlating the process instances with case identifiers, we propose a synthetic log generator namely *synLogGen*, that simulates the event log for a given process profile according to Petri net firing rule. *Process profile* composes activity vocabulary and Petri net. While activity vocabulary holds all valid activity labels in terms of activity type and occurrence priority, Petri net converts the graphed-based reference process model into tabular format. Additionally, unexpected process terminations and

Esgin, E. and Karagoz, P.
Process Profiling based Synthetic Event Log Generation.
DOI: 10.5220/0008363805160524
In *Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2019)*, pages 516-524
ISBN: 978-989-758-382-7

noise effect that deteriorate the business rules are considered as parameters at proposed synthetic event log generation.

The remainder of this paper is organized as follows: The literature review is given in Section 2. In Section 3, we present the details of proposed approach for process log generation, synLogGen. In the following section, experimental analysis and validation with respect to related approaches, i.e. deducing case identifiers (DCI) in (Bayomie et al., 2016) and expectation maximization (E-Max) in (Ferreira & Gillblad, 2009), are analyzed for benchmark business processes given in (Bayomie et al., 2016). Finally, we conclude the paper in Section 5 with a discussion and an outlook on future work.

2 LITERATURE REVIEW

In this section, we summarize the previous efforts on dealing with incomplete labeling in the process event logs. In general, process-aware information systems record a vast array of events without being able to assign them to specific process instances. In such environment, the case identifier attribute is mostly absent and the event log becomes an unlabeled stream of events. In (Bayomie et al., 2016), Bayomie et al. propose an approach to automate the preprocessing step by deducing the case identifiers (DCI) for the unlabeled event log. In addition to the execution log, DCI requires as input the reference process model and heuristic data about the execution timestamp. As a limitation, DCI does not support cyclic processes and inaccurate heuristic data affects the number of possible labeling.

The handling of unlabeled event log is also addressed in (Walicki & Ferreira, 2011) and (Ferreira & Gillblad, 2009). In (Ferreira & Gillblad, 2009), an Expectation-Maximization approach (E-Max) is introduced to estimate a Markov model from an unlabeled event log. It is a greedy heuristic that finds mostly a local maximum of the likelihood function. The main limitation of the corresponding approach is erroneous handling of loops and parallelism in the unlabeled log.

Alternatively, a sequence partitioning approach is presented to produce a set of partitions that represent the minimum cover of the unlabeled log in (Walicki & Ferreira, 2011). Respectively, the underlying approach aims to traverse the complete search space to enumerate all possible solutions within the concept of minimum description length (MDL) stated in (Rissanen, 1978). Similar limitation about handling loops and parallelism is valid for (Walicki & Ferreira, 2011).

There are also other efforts to monitor, and visualize logs, and to map low-level logs to higher-level process tasks. The studies in (Doganata, 2011) and (Doganata & Curbera, 2009) are based on business provenance graph model to generate an automated auditing tool. The main problems are to create internal control points and business artifacts to visualize the process progress. In (Rogge-Solti, 2014) and (Rogge-Solti et al., 2013), a stochastic model is proposed to recommend missing events in the log. They apply path probabilities and Bayesian network to reflect initial beliefs given at reference process model at computing the most probable timestamp. In (Burrett & Gent, 2008), Burrett and Geng propose an iterative workflow mining approach that implements expectation-maximization approach to associate low-level events with high-level tasks.

The proposed work differs from these previous studies such that, rather than handling the labeling problems in the corresponding data set, we propose to generate the event log according to the provided process profile. We especially use the log generation efforts in (Bayomie et al., 2016), (Ferreira & Gillblad, 2009) and (Esgin, Senkul & Cimenbicer, 2010) for comparison.

3 PROPOSED APPROACH

When process mining techniques are applied to the process-aware information systems, we confront the problem about collecting the transactional data, i.e. event log at the source systems. The transactional log at process-aware information systems is not appropriate to monitor the individual business cases. Instead, these systems only monitor the execution log of specific transactions without any relations within a case identifier (van der Aalst, 2006). Indeed, process-aware information systems are strongly data centric: the transactional data is staggered through header and line-item database tables with a many-to-many (*M*:*N*) cardinality. Although SAP tools like Reverse Business Engineering (RBE) log the transaction frequencies, these transactions are linked to event process chain (EPC) format reference process models without any assignment to individual business cases (van der Aalst, 2006).

According to these limitations at correlating the process instances with case identifiers, we propose a synthetic event log generator (*synLogGen*) that simulates the event log from scratch according to

process profile and Petri net firing rule for a given business process.

3.1 Preliminaries on Petri Net

This section introduces the basic Petri net terminology and notation given in (Desel & Esparza, 1995) and (Reisig & Rozenberg, 1998). The classical Petri net is a directed bidirectional graph with two node types: places and transitions. The nodes are connected via directed arcs and connections between two nodes of the same type are restricted (Desel & Esparza, 1995; Reisig & Rozenberg, 1998).

Definition (Petri net). A Petri net is a triple (P, T, F):

- P is a finite set of places. A place p is called an input place of a transition t, i.e. denoted as •t or p•, if and only if there exists a directed arc from p to t otherwise it is called an output place of transition t if and only if there exists a directed arc from t to p, i.e. denoted as t• or •p.
- T is a finite set of transitions ($P \cap T = \emptyset$).
- $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs.

At any time, a place contains any tokens that are denoted as dots (van der Aalst et al., 2003). The state, called as marking, holds the current distribution of these tokens over places and the number of existing tokens varies according to the process structure and successors of the corresponding activity. The marking procedure of Petri net is defined as firing rule as follows (van der Aalst et al., 2003; van der Aalst, 2006):

- A transition t is said to be enabled if and only if each input place contains at least one token.
- An enabled transition may fire. If a transition t is fired, then t consumes one token from each input place p and produces one token from each output place p of t (van der Aalst et al., 2003; van der Aalst, 2006).

3.2 Synthetic Log Generator

Synthetic log generator has a main input which defines the process profile. Process profile composes two data lists: *activity vocabulary* and *Petri net*. Activity vocabulary holds all valid activities in terms of activity type (i.e. I-initiator, O-ordinary, S-sink) and the activity occurrence priority. This priority represents the path probability of corresponding activity occurrence changing at [0, 100] interval and this factor is determined by the domain experts. Petri net converts the graph-based reference process model into tabular format, at which each transition *ti* is listed in terms of predecessor, successor and transition type (i.e. AND/OR/XOR join/split-type gateway or direct succession). Table 1 exemplifies the activity vocabulary and Petri net for the combined xor-and process given in Figure 1.

Table 1: Activity Vocabulary and Petri net for combined xor-and Process.

Activity Vocabulary

Activity ID	Type	Priority
A	I	100%
B	O	50%
C	O	100%
D	O	50%
E	O	100%
F	O	100%
G	O	100%
H	S	100%

Petri net

Transition ID	Predecessor	Successor	Transition type
t1	A	B	*XOR-split*
t2	A	D	*XOR-split*
t3	B	C	*direct succ*
t4	C	E	*XOR-join*
t5	D	E	*XOR-join*
t6	E	F	*AND-split*
t7	E	G	*AND-split*
t8	F	H	*AND-join*
t9	G	H	*AND-join*

According to Petri net firing rule adaptation and input requirements, synthetic event logs are generated as the following steps. Terms in parenthesis refer to the variables and constants addressed in the pseudo-codes Algorithm 1 and 2 respectively.

(1) As the starting point, the initiator (I-typed) activity of the corresponding process (e.g. activity A for combined xor-and process) is tokenized first. Since token list holds just this initiator, the corresponding activity is selected (*selectedActivity*) and fired at fire next activity step. Then a new event log line is created with respect to newly set time stamp, current process instance (*proIns*) and the originator randomly obtained from originator list.
Afterwards, the successors of initiator activity (e.g. activity B and D for the initiator A) are tokenized and added to the token list. According to transition type, and-list and xor-list are also updated.

(2) At the next iteration of fire next activity step, firstly the occupancy of and-list is checked to give priority to any tokenized successors with AND-split transition type. In the case of XOR-split, unfired successors are suppressed at the current process instance (*proIns*). On the other hand, OR-split successors are conditionally fired according to the OR-threshold determined by the domain experts.

(3) If and-list is initial, one of tokenized activities is randomly selected from token list and fired with respect to activity occurrence priority and then a new event log line is created with respect to newly set time stamp, current process instance (*proIns*) and the originator randomly obtained

from originator list. Due to this firing, the successors of fired activity are tokenized.

(4) If any successive activity with AND-join transition type is selected, it is checked whether all predecessors of the corresponding activity are priorly fired and untokenized at fire next activity step (at lines 10-24 at Algorithm 2). According to the tokenized flag, it propagates tokenization to subsequent successors. This tokenization and firing iteration continue up to a sink (S-type) activity is fired (*sinkActivityFired*) or process instance limit (*maxProsIns*) is reached.

In addition to the firing rule, various parameters are used for handling exceptional cases:

- Surprise parameter (SURPRISE_PROB) is used to evoke unexpected process terminations. For instance, bankruptcy is a niche business case in a banking financial process and this relatively least probable case is held by the surprise effect. The default value for surprise probability threshold is 5%.
- Noise parameter (NOISE_FACTOR) is used to generate noisy event log that deteriorates the business rules given in process profile. The default value for noise factor threshold is 5%.
- OR-split gateway specifies that one or more tokenized successors will be fired in the case of OR-split. OR-threshold reflects this conditional firing.

Pseudo-codes for event log generation and fire next activity steps are given in Algorithm 1 and Algorithm 2, respectively.

Algorithm 1: Generate Logs(*actListFile, pnFile, maxProIns*).

```
1:  activityList ← Read Activity List(actListFile)
2:  petriNet ← Read Petri Net(pnFile)
3:  proIns ← 1
4:  fireCount ← 0
5:  while proIns ≤ maxProIns do
6:    surprise ← SURPRISE_PROB
7:    Set sinkActivityFired as FALSE
8:    currActivity ← Get Initiator Activity(activityList)
9:    Set tokenized attribute of currActivity as TRUE
10:   Add currActivity to tokenList
11:   while surprise ≥ SURPRISE_PROB AND
      !sinkActivityFired do
12:     Fire Next Activity(proIns)        // Algorithm 2
13:     Random Generate surprise from [0, 100]
14:     if surprise < SURPRISE_PROB then
15:       Reset Lists(andList, xorList, tokenList)
16:       Reset Attributes(tokenized, fired) of all activity at
      activityList
17:     endif
18:   endwhile
19:   proIns++
20: endwhile
```

Algorithm 2: Fire Next Activity(*proIns*).

```
1:  Set getFired as FALSE
2:  while !getFired do
3:    if andList IS NOT INITIAL then
4:      selectedActivity ← Random Select Activity wrt
      priority attribute from andList
5:    else
6:      selectedActivity ← Random Select Activity wrt
      priority attribute from tokenList
7:    endif
8:    Get originator from originatorList
9:    if predecessorList of selectedActivity IS NOT
      INITIAL then
10:     tokenized ← Check Tokenization at predecessorList
      of selectedActivity
11:     Random Generate noise from [0, 100]
12:     if !tokenized AND noise < NOISE_FACTOR then
13:       fireCount++
14:       Set timestamp wrt fireCount
15:       Generate newEventLog wrt {timestamp,
      originator, selectedActivity, proIns}
16:       Set tokenized attribute of selectedActivity as
      FALSE
17:       Set fired attribute of selectedActivity as TRUE
18:       Refresh Lists(andList, xorList, tokenList) wrt
      selectedActivity and its successorList
19:       Set getFired as TRUE
20:       if selectedActivity IS SINK then
21:         Set sinkActivityFired as TRUE
22:         Reset Lists(andList, xorList, tokenList)
23:         Reset Attributes(tokenized, fired) of all activity
      at activityList
24:       endif
25:     endif
26:   else
27:     fireCount++
28:     Set timestamp wrt fireCount
29:     Generate newEventLog wrt {timestamp,
      originator, selectedActivity, processInstance}
30:     Set tokenized attribute of selectedActivity as
      FALSE
31:     Set fired attribute of selectedActivity as TRUE
32:     Refresh Lists(andList, xorList, tokenList) wrt
      selectedActivity and its successorList
33:     Set getFired as TRUE
34:     if selectedActivity IS SINK then
35:       Set sinkActivityFired as TRUE
36:       Reset Lists(andList, xorList, tokenList)
37:       Reset Attributes(tokenized, fired) of all activity
      at activityList
38:     endif
39:   endif
40: endwhile
```

4 EXPERIMENTAL ANALYSIS AND DISCUSSION

As benchmark process models, we use the xor, and, combined xor-and, and nested xor-and processes, which are also referenced in (Bayomie et al., 2016). According to the process models given in Figure 1, process profiles are prepared and synthetic event log

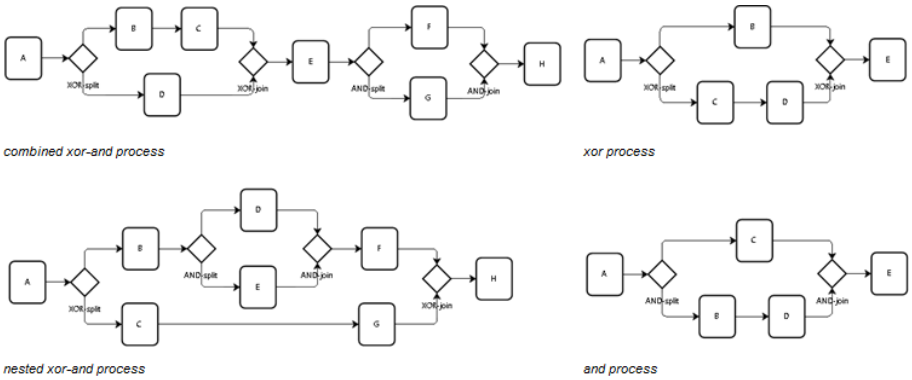

Figure 1: Benchmark Processes for Evaluation as referenced in (Bayomie et al., 2016).

are generated according to given number of cases (maxProIns) as given in Table 2.

Table 2: Benchmark Process Characteristics according to Candidate (DCI and E-Max) and Proposed Approach synLogGen.

Benchmark Process	Criteria	DCI	E-Max	synLogGen
combined xor-and process	*Number of Cases*	100	104	102
	Log Size	651	651	627
nested xor-and process	*Number of Cases*	100	149	125
	Log Size	498	498	624
xor process	*Number of Cases*	100	103	103
	Log Size	353	353	354
and process	*Number of Cases*	100	201	201
	Log Size	500	500	978

As given in Section 4.2, we use two previous studies, DCI (Bayomie et al., 2016) and E-Max (Ferreira & Gillblad, 2009) that priorly worked on the benchmark processes in (Bayomie et al., 2016), for performance comparison. For this reason, we present the process log generation parameters of these works in Table 2. The basics of these two studies are as follows:

- DCI (deducing case identifiers) generates a set of labeled event log by considering all labeling possibilities for each activity in the event log (Bayomie et al., 2016). Additionally, a ranking score that indicates the degree of trust in labeling of events within each log is assigned (Bayomie et al., 2016).
- E-Max (Expectation-Maximization) is a greedy algorithm that mines the unlabeled event log (Ferreira & Gillblad, 2009). It takes the local maximum to generate one labeled event log.

4.1 Analysis on Compatibility of Event Logs with Intended Process Model

After the event logs are generated by our solution, in order to analyze the compatibility of the generated logs with the target model, we extract the process back from the generated logs and compare with the target model. We used the process mining approach given in (Esgin, Senkul & Cimenbicer, 2010). The underlying process discovery approach is composed of two components: from-to chart and process flow branch discovery. From-to chart is an analytical tool, which is basically used in monitoring material handling routes between operations, machines, departments or work centers on the production floor (Esgin & Senkul, 2009). It is adapted to monitor transitions among activities occurred in process instances and to figure out if there exists any specific order of the occurrences for representing in process model (Esgin, Senkul & Cimenbicer, 2010).

We aim to compare the quality of mined process models by *completeness* and *soundness* metrics, which are defined as follows:

- Completeness of the process model is the ratio of the traces in the event log that may be the result of some enactment at the corresponding process model (van Dongen, Dijkman & Mendling, 2008). The completeness metric is similar to fitness defined in (Jagadeesh Chandra Bose & van der Aalst, 2012), (Rozinat & van der Aalst, 2008) and recall defined in (Becker & Laue, 2012) and (Gerke, Cardoso & Claus, 2008).
- Soundness measures the ratio of the activity enactments within the corresponding process model that find some correspondence in the event logs (van Dongen, Dijkman & Mendling, 2008). The soundness metric is similar to minimality or behavioral appropriateness defined in (Maruster, Weijters & van der Bosch, 2006) and precision defined in (Jagadeesh Chandra Bose & van der Aalst, 2012), (Becker & Laue, 2012) and (Gerke, Cardoso & Claus, 2008).

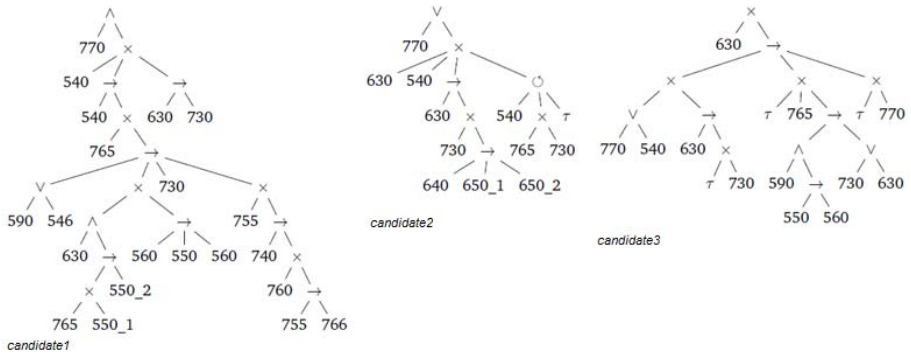

Figure 3: Process Maps per Candidate Process Variant (*candidate*$_i$) as referenced in (Yilmaz & Karagoz, 2015), (Buijs & Reijers, 2014) and (van der Aalst, 2011).

As shown in Figure 2, the combination of proposed approach synLogGen with the process discovery approach introduced in (Esgin, Senkul & Cimenbicer, 2010) captures the process behavior at a high accuracy degree, having an average completeness value over 95%. Respectively, XOR gateways at nested xor-and process results in a more spaghetti-like process model with a 38.8% soundness value.

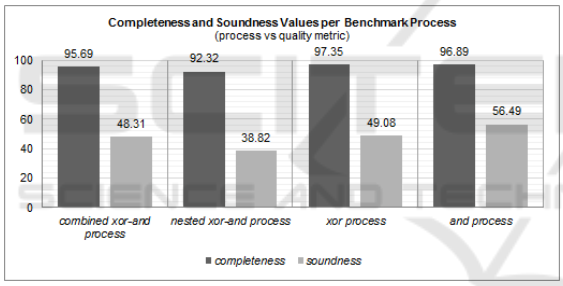

Figure 2: Completeness and Soundness Values per Benchmark Process (according to proposed approach synLogGen).

In addition to the benchmark process models referenced in (Bayomie et al., 2016), we aim to analyze the performance of proposed approach at handling relatively complex business processes. Hence the proposed approach synLogGen is applied and evaluated on the Environmental Permit Application business process in Configurable Services for Local Governments (CoSeLoG) project, which investigates the similarities and deviations between processes of different municipalities in Netherlands (Yilmaz & Karagoz, 2015; Buijs & Reijers, 2014; van der Aalst, 2011).

According to the process maps of candidate process variants given in Figure 3, process profiles are designated, and event logs are generated according to varying noise factor, i.e. noise in [%0.5, %10] value range. Then process discovery is performed for each noisy event log dataset.

As shown in Figure 4 and 5, the combination of proposed approach synLogGen with the process mining algorithm introduced in (Esgin, Senkul & Cimenbicer, 2010) is robust to the noise factor effect at event log generation and process discovery. According to completeness metric, complex candidate process variants with relatively deeper process maps (i.e. candidate1 figured within 10-level and candidate3 within 7-level process map) have a %1.46 average loss per noise factor increment, while candidate2 figured within a lasagna-like structured process map has a %0.65 average loss.

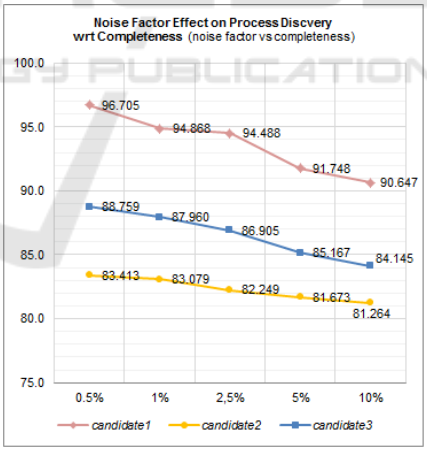

Figure 4: Noise Factor Effect on Process Discovery with respect to Completeness. Although spaghetti-like complex process variants with deeper process maps are potentially more vulnerable to the increments at noise factor in the first glance, a %1.46 average loss at completeness values implies a relatively robust approach that appropriately extracts relatively dominant process behaviors at a noisy process history.

Similar outcomes are valid for soundness metric such that, while candidate1 and candidate3 have a %1.36 average loss per noise factor increment,

candidate2 performs a mitigated average loss of %0.43.

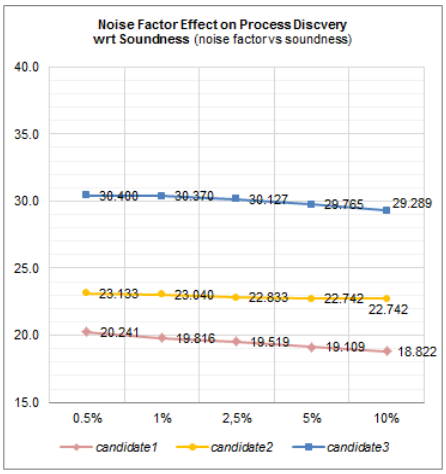

Figure 5: Noise Factor Effect on Process Discovery with respect to Soundness.

4.2 Log Generation Analysis in Comparison to the Literature on Compatibility of Event Logs with Intended Process Model

Since the process event logs generated by DCI and E-Max in (Bayomie et al., 2016) are not available, we follow an indirect way of performance comparison. Therefore, process discovery performance is evaluated according to mined Petri nets given in (Bayomie et al., 2016). Although the accuracy of the discovered process depends on both log generation and process discovery method used, it gives a hint as to the quality of the generated logs.

As the accuracy metric, dissimilarity metric (*dissim*) introduced in (Esgin & Senkul, 2011) is applied to measure the graph-based structural deviance between reference process models and mined process behaviors. Underlying dissimilarity metric is built on a vector model from information retrieval and an abstraction of process behavior called process triple. Process triple is a set that covers activity existence and interactions in terms of successors and predecessors among activities (Esgin & Senkul, 2011).

According to dissimilarity values given in Figure 6, DCI and the combination of proposed approach synLogGen with the process discovery approach in (Esgin, Senkul & Cimenbicer, 2010) are respectively more accurate in distilling the process behavior given in the reference process models than E-Max. While average dissimilarity value of E-Max is approximately 0.554, the average values of both DCI and proposed approach synLogGen highlight less discrepancies between the intended process behaviors given at reference process model and discovered behaviors (i.e. 0.151 and 0.14 respectively).

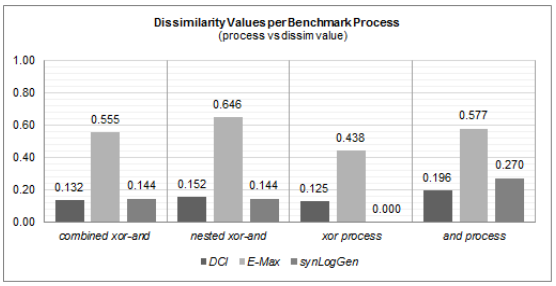

Figure 6: Dissimilarity Values per Benchmark Process. Higher dissimilarity value implies a strong distinction between reference and mined process model (i.e. 0: exactly the same, 1: quite distinct).

In addition to dissimilarity metric, we aim to analyze the understandability of mined process models in terms of connectivity. *Connectivity* is the average number of transitions (|T|) per activity (|A|) at the corresponding process model (|T|/|A|). As shown in Figure 7, candidate approaches, i.e. DCI and E-Max, discover more spaghetti-like Petri nets with higher connectivity values. Alternatively, proposed approach synLogGen discovers more structured and lasagna-like process models according to less deviation with reference process model.

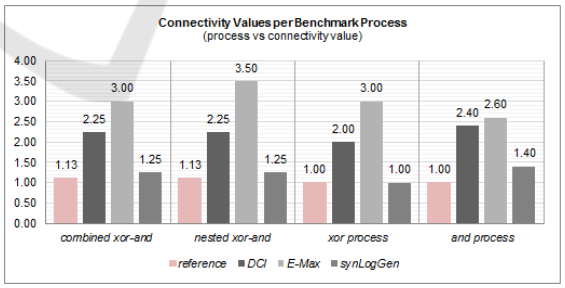

Figure 7: Connectivity Values per Benchmark Process. While higher connectivity values of candidate approaches, DCI and E-Max, imply spaghetti-like process model discovery, less deviance between reference and proposed apparoach synLogGen emphasizes more structured process models.

5 CONCLUSION AND FUTURE WORK

Process-aware information systems record detailed process execution information concerning the processes they support. Typically, the assignment of case identifiers to the process instances is missing, due to data centric architecture of the underlying information systems (van der Aalst, 2006). In this paper, we introduce an approach to simulate synthetic event log from scratch rather than repairing incomplete unlabeled event log. We use as input the process profile that defines the activity vocabulary, i.e. a list of valid activity labels, types and occurrence priority, and Petri net in tabular form. Proposed approach synLogGen also takes both noise and surprise effects into account at applying Petri net firing rule. According to completeness and soundness outcomes obtained at the experimental runs with varying noise factor values, the combination of proposed approach synLogGen with the process discovery approach introduced in (Esgin, Senkul & Cimenbicer, 2010) is robust to the corresponding noise effect at distilling the process behaviors in a relatively noisy process execution environment.

In the evaluation step, four process models that are referenced in (Bayomie et al., 2016) are determined as benchmark processes. Two prior approaches, i.e. DCI given in (Bayomie et al., 2016) and E-Max given in (Ferreira & Gillblad, 2009), which priorly handled the corresponding benchmark processes, are selected as candidate. According to the accuracy aspect, DCI and the combination of proposed approach synLogGen with the process discovery approach introduced in (Esgin, Senkul & Cimenbicer, 2010) are respectively more accurate than E-Max in terms of dissimilarity metric (Esgin & Senkul, 2011), which measures the discrepancies between reference process model and mined process behaviors on a graph-based structural similarity measurement. Additionally, proposed approach synLogGen generates more structured and lasagna-like process models with respect to moderate connectivity values and less variance with reference process model.

As the future work, we intend to implement an integer-programming (IP) based synthetic log generation, in which the process behaviors at reference process model and the activity frequencies at the unlabeled event log act as the main constraints. The corresponding IP-based approach will simulate maximal process traces without any relaxations at these constraints.

REFERENCES

Bayomie, D., Helal, I. M. A., Awad, A., Ezat, E., & Elbastawissi, A. (2016). Deducing Case IDs for Unlabeled Event Logs. *International Conference on Business Process Management*, vol. 256, 242-254.

Becker, M., & Laue, R. (2012). A Comparative Survey of Business Process Similarity Measures. *Computers in Industry*, vol. 63(2), 148-167.

Buijs, J.C., & Reijers, H.A. (2014). Comparing Business Process Variants Using Models and Event Logs. *Enterprise, Business-Process and Information Systems Modeling*, 154-168.

Burrett, S., & Geng, L. (2008). Bayesian Classification of Events for Task Labeling Using Work-Flow Models. *Proceedings of the 4th Workshop on Business Process Intelligence*.

Desel, J. & Esparza, J. (1995). Free Choice Petri Nets. *Cambridge Tracts in Theoretical Computer Science*, vol. 40, Cambridge University Press, Cambridge, UK.

Doganata, Y. N. (2011). Designing Internal Control Points in Partially Managed Processes by Using Business Vocabulary. *ICDE Workshops*. 267–272.

Doganata, Y. N., & Curbera, F. (2009). Effect of Using Automated Auditing Tools on Detecting Compliance Failures in Unmanaged Processes. *BPM. LNCS*, vol. 5701, 310-326.

Dustdar, S., & Gombotz, R. (2006). Discovering Web Service Workflows Using Web Services Interaction Mining. *International Journal of Business Process Integration and Management*, vol. 1(4), 256.

Esgin, E., & Senkul, P. (2009). Hybrid Approach to Process Mining: Finding Immediate Successors of a Process by Using From-to Chart. *International Conference on Machine Learning and Applications*, 664-668.

Esgin, E., & Senkul, P. (2011). Delta Analysis: A Hybrid Quantitative Approach for Measuring Discrepancies between Business Process Models. *6th International Conference, Hybrid Artificial Intelligence Systems (HAIS)*, vol. 6678, 296-304.

Esgin, E., Senkul, P., & Cimenbicer, C. (2010). A Hybrid Approach for Process Mining: Using From-to Chart Arranged by Genetic Algorithms. *5th International Conference, Hybrid Artificial Intelligence Systems (HAIS)*, vol. 6076, 178-186.

Ferreira, D. R., & Gillblad, D. (2009). Discovering Process Models from Unlabelled Event Logs. *BPM LNCS*, vol. 5701, 143–158.

Gerke, K., Cardoso, J., & Claus, A. (2009). Measuring the Compliance of Processes with Reference Models. On *the Move to Meaningful Internet Systems: OTM 2009, Confederated International Conferences: CoopIS, IS, DOA and ODBASE*, vol. 5870, 76-93.

Gunther, C. W., & van der Aalst, W. M. P. (2006). *Process Mining in Case Handling Systems*. Multikonferenz Wirtschaftsinformatik 2006.

Jagadeesh Chandra Bose, R. P., & van der Aalst, W. M. P. (2012). *When Process Mining Meets Bioinformatics*.

CAISE Forum, IS Olympics: Information Systems in a Diverse World, vol. 107, 202-217.

Maruster, L., Weijters, T., & van der Bosch, A. (2006). A Rule-Based Approach for Process Discovery. *Data Mining and Knowledge Discovery*, vol. 13(1), 67-87.

Reisig, W. & Rozenberg, G. (1998). *Lectures on Petri Nets I: Basic Models*, vol. 1491 LNCS.

Rissanen. J. (1978). Modeling by Shortest Data Description. *Automatica*, vol. 14 (5), 465–471.

Rogge-Solti, A. (2014). *Probabilistic Estimation of Unobservered Process Events*. Phd, University of Potsdam.

Rogge-Solti, A., Mans, R., van der Aalst, W. M. P. & Weske, M. (2013). Repairing Event Logs Using Timed Process Models. *OTM Workshops*. LNCS, vol. 8186, 705–708.

Rozinat, A., & van der Aalst, W. M. P. (2008) Conformance Checking of Processes Based on Monitoring Real Behavior. *Information Systems*, vol. 33(1), 64-95.

van der Aalst, W. M. P. (2006). Matching Observed Behavior and Modeled Behavior: An Approach Based on Petri nets and Integer Programming. *Decision Support Systems*, vol. 42(3), 1843-1859.

van der Aalst W. M. P. (2011). Process Mining Manifesto. *BPM Workshops. LNBIP*, vol. 99, 169–194.

van der Aalst, W. M. P. (2011). Intra and Inter-Organizational Process Mining: Discovering Processes within and between Organizations. *IFIP Working Conference on the Practice of Enterprise Modeling*, vol. 92, 1-11.

van der Aalst, W. M. P., Dongen, B. F., Herbst, J., L., M., Schimm, G., & Weijters, T. A. J. M. M. (2003). Workflow Mining: A Survey of Issues and Approaches. *Data & Knowledge Engineering*, vol. 47(2), 237-267.

van der Aalst, W. M. P., Gunther, C., Recker, J., & Reichert, M. (2006). Using Process Mining to Analyze and Improve Process Flexibility. *BPMDS 2006*.

van Dongen, B., Dijkman, R., & Mendling, J. (2008). Measuring Similarity between Business Process Models. *Advanced Information Systems Engineering, 20th International Conference, CAISE*, vol. 5074, 450-464.

Walicki, M. & Ferreira, D. R. (2011). Sequence Partitioning for Process Mining with Unlabeled Event Logs. *Data & Knowledge Engineering*, vol. 70(10), 821–841.

Weijters, T. A. J. M. M., & van der Aalst, W. M. P. (2003). Rediscovering Workflow Models from Event-Based Data Using Little Thumb. *Integrated Computer-Aided Engineering*, vol. 10(2), 151-162.

Yilmaz, O., & Karagoz, P. (2015). Generating Performance Improvement Suggestions by using Cross-Organizational Process Mining. *5th International Symposium on Data-Driven Process Discovery and Analysis*, 3-17.