An Empirical Evaluation of Process Mining Algorithms based on Reference Models

Jianmin Wang[1], Lijie Wen[1], Shijie Tan[1], and Haiping Zha[2]

[1] School of Software, Tsinghua University, Beijing 100084, China
[2] Institute of Specifications and Standards, Shanghai 200235, China
jimwang@tsinghua.edu.cn, wenlj00@mails.tsinghua.edu.cn

Abstract. In this paper, through adapting the evaluation framework proposed by Rozinat et al., an empirical study on evaluating the process mining algorithms based on reference models is conducted, in which the quality of a discovered model is measured by the behavior and structure similarities with its reference model. In our empirical study, artificial reference models with typical control-flow constructs(e.g., short-loop, non-free choice) are collected from academic papers and SAP suites, and the real-life reference models from an industrial boiler manufacturer are systematically reviewed. Through our empirical study, some well-known results, such as a certain process mining algorithm is good at dealing with a kind of process models with a typical structural feature, are validated. Besides, we find several other interesting facts, (1) comparing with the reference models, the evaluation results are more understandable, especially for the models with deficiencies, (2) some reference models, which have significant power to differentiate the process mining algorithms, are also discovered, (3) process mining can be used to improve, even to correct, some of the real-life process models.

1 Introduction

The pioneering efforts towards a common evaluation framework for process mining algorithms were done by Rozinat et al. recently[7]. In the evaluation framework, there are four elements, (1)event log and reference model repository, (2)log generation module, (3)modification and verification tools, (4)evaluation and comparison module. Just as the authors point out[7], a comprehensive set of benchmark examples (ideally containing both artificial and real-life process models) is still lacking.

In this paper, through adapting the evaluation framework by Rozinat et al.[7], we do an empirical study on evaluating process mining algorithms by measuring the behavior similarity and the structure similarity between the reference process models (user knowledge) and the mined models (discovered knowledge).

In order to carry out the empirical study, we build an process mining algorithm evaluation module (call it evaluation system thereafter) as well as its reference models in the BeehiveZ project[3]. In our empirical study, artificial reference

[3] http://code.google.com/p/beehivez/downloads/list

models with typical control-flow constructs (e.g., short-loop, non-free choice) are collected from academic papers and SAP suites, and the real-life reference models from an industrial boiler manufacturer are systematically reviewed. All the reference models are grouped into categories with some kinds of control-flow features, e.g., short-loop, non-free-choice, in the reference model repository.

The contributions of this paper are as follows. (a) We build an evaluation system with extensible architecture. (b) We build an exemplary reference model repository consisting both artificial and real-life reference models. (c) Through our empirical study, some well-known results, such as a certain process mining algorithm is good at dealing with a kind of process models with a typical structural feature, are validated. Besides, we find several other interesting facts, (1) comparing with the reference models, the evaluation results are more understandable, especially for the worse mined models, (2) some reference models, which have significant power to differentiate the process mining algorithms, are also discovered, (3) process mining can be used to improve, even to correct, some of the real-life process models.

The remainder of this paper is organized as follows. Section 2 reviews the related work. Section 3 presents the overall architecture and essential elements of our evaluation system. Section 4 shows the experiments to evaluate process mining algorithms using the benchmark framework. Section 5 concludes the paper and outlines future work.

2 Related work

Process mining aims to discover exact process models from event logs. In [10], the α algorithm, which can construct the final process model in WF-nets from event logs, was proposed. The strengths and weaknesses of the α algorithm had been well studied. Therefore, [10] provides a good start point for further research. After that, a lot of process mining approaches were proposed [3, 4, 15].

At the same time, different tools for process mining are also presented, such as Little Thumb [13], EMiT [12], Process Miner [9], and InWoLvE [4]. These tools are intended for special process mining algorithms. The ProM framework [11] is a common framework to facilitate process mining implementation. Many process mining algorithms have been implemented as plugins in the ProM framework which are available for public.

Most researchers have been aware of the importance of performance evaluation of the algorithms, they proposed their own criterions along with their mining algorithms, such as completeness, irredundancy, and minimality [1], or fitness, structural appropriateness, and behavioral appropriateness [8].

The pioneering efforts towards a common evaluation framework for process mining have been done in [7]. The proposal focuses on the test of compliance between the mined process model and the events logs from four dimensions, i.e., fitness, precision, generalization and structure. Being complementary of [7], our empirical study focuses on the benchmark based on the reference models.

3 Evaluation System

3.1 Overall architecture

The architecture of the evaluation system is shown in Figure 1. It includes five key functional elements, i.e., *log generator*, *process miner*, *similarity measurer*, *conformance checker* and *result navigator*.

The function of the log generator is to automatically generate event logs of a process model. The event log is the input of the process miner which invokes the tested process mining algorithm by a unified interface. The mining algorithm will rediscover process models from the event log. Finally, the similarity measurer will invoke the similarity algorithm to calculate the similarity between the original process model and the mined process model. Also, the conformance checker will calculate conformance metrics between the event log and the mined model. At the end, the evaluation results will be shown in the result navigator visually.

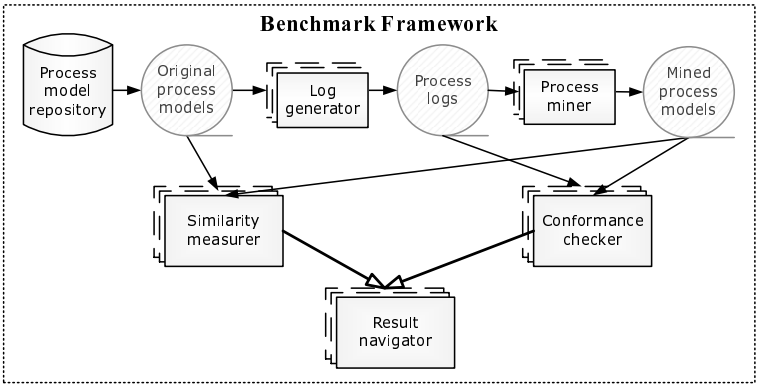


Fig. 1. The architecture of the evaluation system

3.2 Process model repository

Process model repository consists of tow kinds of reference models, artificial reference models and real-life reference models.

All process models in the process model repository are organized in categories according to their characteristics. Concerning the typical characteristics of control flows, we pre-define the following categories, *simple sequence* (*seq*), *simple selection* (*sel*), *simple parallel* (*par*), *short loop* (*slp*), *non-free-choice* (*nfc*), *invisible task* (*inv*), *duplicate task* (*dup*), *nested loop* (*nlp*) and *non-WF-net* (*nwf*). Furthermore, the process model repository supports customizable categories.

The process model repository includes 70 artificial models now divided into 7 categories, whose sources are listed in Table 1. For each category, there are 10 process models modeled in WF-nets. Totally there are three kinds of sources for these models, i.e., from papers , by hand and converted from SAP reference models. The structural characteristics of these 70 models are illustrated in 2 by the metrics derived and borrowed from [6].

Table 1. Sources of 70 artificial process models in the process model repository

Category	*seq*	*par*	*sel*	*slp*	*nfc*	*inv*	*dup*
From Papers	0	0	0	8 [2]	10 [15]	10 [16]	7 [5]
From SAP	6	7	10	0	0	0	0
By Hand	4	3	0	2	0	0	3

The metrics in Table 2 are introduced as follows: S_T, S_P and S_F for numbers of transitions, places and arcs respectively; $S_{S_{AND}}$, $S_{J_{AND}}$, $S_{S_{XOR}}$ and $S_{J_{XOR}}$ for numbers of AND-Split, AND-Join, XOR-Split and XOR-Join respectively; $diam$ for the length of the longest path from a start node to an end node; Δ for the number of arcs divided by the number of the maximum number of arcs for the same number of nodes; CNC for the ratio of arcs to nodes; $\widehat{d_C}$ for the maximum degree of a connector; Π for the number of cut-vertices divided by the number of nodes; Ξ for the number of arcs between none-connector nodes divided by the number of arcs; Φ for one minus the number of nodes in the reduced model divided by the number of nodes in the original model; Λ for the maximum depth of all nodes; MM for the sum of mismatches for each connector type; CFC for the sum over all connectors weighted by their potential combinations of states after the split; CYC for nodes on cycles divided by the number of nodes; TS for the sum of the output-degree of AND-Splits minus one.

Table 2. Mean and Standard Deviation of 70 artificial models disaggregated by group

Metric	*seq*		*par*		*sel*		*slp*		*nfc*		*inv*		*dup*	
	μ_1	σ_1	μ_2	σ_2	μ_3	σ_3	μ_4	σ_4	μ_5	σ_5	μ_6	σ_6	μ_7	σ_7
S_T	4.4	2.17	5.8	1.62	6.6	2.41	5.8	1.55	6.4	1.26	5.2	0.92	6.9	2.73
S_P	5.4	2.17	8.3	2.21	5.4	1.96	5.1	0.99	7.3	1.16	4.9	0.88	7.8	3.12
S_F	8.8	4.34	14.6	4.43	13.2	4.83	11.6	3.10	16.8	3.16	10.8	1.93	15.2	7.07
$S_{S_{AND}}$	0.0	0.00	1.2	0.42	0.0	0.00	0.0	0.00	2.0	0.47	0.2	0.42	0.7	0.95
$S_{J_{AND}}$	0.0	0.00	1.2	0.42	0.0	0.00	0.0	0.00	2.0	0.47	0.2	0.42	0.7	0.95
$S_{S_{XOR}}$	0.0	0.00	0.0	0.00	1.2	0.42	1.7	0.82	2.1	0.74	1.3	0.67	0.7	0.67
$S_{J_{XOR}}$	0.0	0.00	0.0	0.00	1.7	0.67	1.7	0.82	2.1	0.74	1.2	0.63	0.7	0.67
$diam$	8.8	4.34	8.2	2.20	8.0	3.27	6.6	0.97	8.4	2.63	6.8	1.69	9.4	2.12
Δ	0.2	0.09	0.2	0.04	0.2	0.07	0.2	0.04	0.2	0.04	0.2	0.03	0.2	0.04
CNC	0.9	0.06	1.0	0.04	1.1	0.12	1.1	0.06	1.2	0.08	1.1	0.07	1.0	0.09
$\overline{dc}$	0.0	0.00	2.3	0.42	2.4	0.63	2.0	0.00	2.0	0.00	2.2	0.42	1.5	1.08
$\widehat{dc}$	0.0	0.00	2.3	0.48	3.0	1.49	2.0	0.00	2.0	0.00	2.3	0.48	1.5	1.08
Π	1.0	0.00	0.3	0.09	0.5	0.20	0.6	0.18	0.2	0.12	0.4	0.25	0.6	0.36
Ξ	1.0	0.00	0.8	0.03	0.8	0.05	0.7	0.09	0.5	0.10	0.7	0.10	0.9	0.11
Φ	0.4	0.05	0.3	0.08	0.3	0.16	0.3	0.16	0.1	0.14	0.3	0.13	0.3	0.09
Λ	0.0	0.00	1.0	0.00	1.0	0.00	1.0	0.00	1.7	0.48	1.2	0.42	1.0	0.82
MM	0.0	0.00	0.0	0.00	0.5	0.71	0.0	0.00	0.0	0.00	0.1	0.32	0.0	0.00
CH	0.0	0.00	0.0	0.00	0.0	0.00	0.0	0.00	1.0	0.00	0.2	0.42	0.3	0.46
CFC	0.0	0.00	1.2	0.42	3.5	1.58	3.4	1.65	6.2	1.93	3.0	1.25	2.2	2.04
CYC	0.0	0.00	0.0	0.00	0.0	0.00	0.4	0.10	0.1	0.2	0.1	0.23	0.0	0.07
TS	0.0	0.00	1.5	0.71	0.0	0.00	0.0	0.00	2.0	0.47	0.2	0.42	0.7	0.95

In the repository, there are 57 real-life process models collected from a boiler manufacturer in the southeast of China, which produces the large-scale industrial boilers consisting of more than 10,000 parts and assemblies. These real-life models are converted from EPC models designed by a third-party consulting company. These models are scattered in five SAP modules (i.e., *CO* for Cost

Table 3. Distribution of 57 real-life process models in the process model repository

Category	*dup*	*inv*	*slp*	*nlp*	*nwf*
CO	0	8	3	2	5
FA	1	8	1	0	11
MM	1	3	0	0	6
PP	0	1	0	0	2
SD	0	3	0	0	2

Controlling, *FA* for Financial Accounting, *MM* for Material Management, *PP* for Production Planning, *SD* for Sales and Distribution) are organized into 6 categories, whose distributions are shown in Table 3. The structural characteristics of these 57 models are illustrated in Table 4.

Table 4. Mean and Standard Deviation of 57 real-life models disaggregated by group

Metric	*dup*		*inv*		*slp*		*nlp*		*nwf*	
	μ_1	σ_1	μ_2	σ_2	μ_3	σ_3	μ_4	σ_4	μ_5	σ_5
S_T	10.0	0.00	8.0	2.16	7.8	1.50	7.0	0.00	7.7	2.43
S_P	10.5	0.71	7.8	1.56	7.3	1.89	6.0	0.00	8.1	1.79
S_F	21.0	1.41	16.2	4.46	16.0	4.00	14.0	0.00	15.6	4.98
$S_{S_{AND}}$	0.5	0.71	0.2	0.42	0.3	0.50	0.0	0.00	0.2	0.43
$S_{J_{AND}}$	0.0	0.00	0.0	0.21	0.0	0.00	0.0	0.00	0.0	0.2
$S_{S_{XOR}}$	1.5	0.71	1.8	0.83	1.8	0.50	2.0	0.00	1.8	1.07
$S_{J_{XOR}}$	1.0	1.41	1.1	0.79	1.3	0.50	1.0	0.00	0.7	0.87
$diam$	8.0	2.83	10.3	3.68	8.5	3.00	10.0	0.00	8.8	3.76
Δ	0.1	0.01	0.1	0.03	0.2	0.03	0.2	0.00	0.1	0.03
CNC	1.0	0.10	1.0	0.07	1.1	0.04	1.1	0.00	1.0	0.07
$\overline{dc}$	2.3	0.35	2.1	0.22	2.3	0.21	2.3	0.00	2.0	0.49
$\widehat{dc}$	2.5	0.71	2.4	0.58	2.8	0.50	3.0	0.00	2.2	0.67
Π	0.6	0.48	0.6	0.23	0.4	0.15	0.4	0.00	0.7	0.20
Ξ	0.9	0.06	0.8	0.03	0.8	0.02	0.8	0.00	0.8	0.07
Φ	0.4	0.01	0.3	0.14	0.2	0.13	0.2	0.00	0.2	0.14
Λ	0.5	0.71	0.8	0.58	0.8	0.50	1.0	0.00	0.5	0.65
MM	2.0	0.00	1.0	0.90	1.3	0.50	1.0	0.00	1.3	0.68
CH	0.4	0.57	0.2	0.38	0.2	0.41	0.0	0.00	0.2	0.37
CFC	3.5	0.71	4.1	1.96	3.8	0.50	4.0	0.00	4.0	2.35
CYC	0.1	0.14	0.2	0.27	0.5	0.28	0.8	0.00	0.1	0.20
TS	1.0	1.41	0.3	0.54	0.5	1.00	0.0	0.00	0.2	0.43

3.3 Log generator

A challenging problem of artificial event logs is how to meet the requirement of *completeness* as to control flow discovery. Different process mining algorithms have different requirements on completeness of event logs. By definition, complete event logs of a process model should record all observed behavior of the process model. However, many processes contain loop constructs, their full firing sequences are infinite. Therefore, the behavior of such a process model cannot be completely recorded in a finite set of event logs.

To cope with the above challenge, we take two strategies in the implementation of our evaluation system. Firstly, the event log of a process model is generated based on the fairness principle, i.e., each enabled transition has the

same probability to be fired in each step. So that, each possible execution path of the process can be covered with equal probabilities. Secondly, depending on a user-defined value for TAR (Task Adjacency Relation) [17] completeness of the log being generated, the generation procedure will terminate automatically.

3.4 Similarity measurer

In our empirical study, we just use the TAR similarity[17] for calculating the average behavior similarity. As to the structure similarity, we use the Context similarity (CON for short) for calculating the average structure similarity. The core idea of CON similarity is that the similarity between two nodes depends not only on themselves but also on the context of them, i.e., the similarities between their input and output nodes respectively.

The evaluation result of a process mining algorithm depends obviously on the precision of the process similarity measures. In order to mitigate the impact of the similarity deviations, we use the behavior and structure similarities together. Moreover, in our empirical study, we pay more attention to the similarity outliers, which are far less than the upper bond (e.g. 1.0) and are more significant to find the problem of the process mining algorithm.

4 Experimental evaluation

We use our evaluation system to evaluate process mining algorithms which are available in ProM. Seven typical process mining algorithms, i.e., the α algorithm [10], the α^{++} algorithm [15], the $\alpha^{\#}$ algorithm [16], the genetic algorithm (GA for short) [3], the duplicate genetic algorithm (DGA for short) [3], the heuristics miner (Heu for short) [14], and the region miner (Reg for short), are selected. The results from theoretical analysis on the mining capabilities of them are listed in Table 5 (+ for support; – for not support; × for forbidden; $\surd$ for required; none for not required; DS, DS+ and DS++ for direct succession, two-step closure succession and long-term succession respectively; ES for event significance; TS for trace significance and GC for global completeness).

Table 5. Theoretical analysis on the mining capabilities of 7 mining algorithms

Features	*seq*	*par*	*sel*	*slp*	*nfc*	*inv*	*dup*	*nlp*	*nwf*	*noise*	*completeness*	*frequency*
α	+	+	+	+/–	–	–	–	+	+	×	DS	none
α^{++}	+	+	+	+	+	–	–	+	+	×	DS++	none
$\alpha^{\#}$	+	+	+	+	–	+	–	+	+	×	DS+	none
GA	+	+	+	+	+	+	–	+	+	none	TS	$\surd$
DGA	+	+	+	+	+	+	+	+	+	none	TS	$\surd$
Heu	+	+	+	+	+/–	+	–	+	+	none	ES	$\surd$
Reg	+	+	+	+	+	–	–	+	+	×	GC	none

4.1 Empirical study on artificial models

The benchmark results based on 70 artificial models are listed in Table 6, from which several common senses can be observed. Firstly, almost all algorithms have

no difficulties in dealing with sequence, parallel, selection and short loop. Secondly, the $\alpha^{\#}$ algorithm has the best performance on invisible tasks. Thirdly, the α^{++} algorithm has the best performance on non-free-choice construct. Fourthly, the DGA algorithm has the best performance on duplicate tasks. Fifthly, the $\alpha^{\#}$ algorithm and the Region miner perform very well on duplicate tasks considering behaviors. However, both of them get low structure similarities because they construct too many invisible tasks to imitate the behaviors of duplicate tasks.

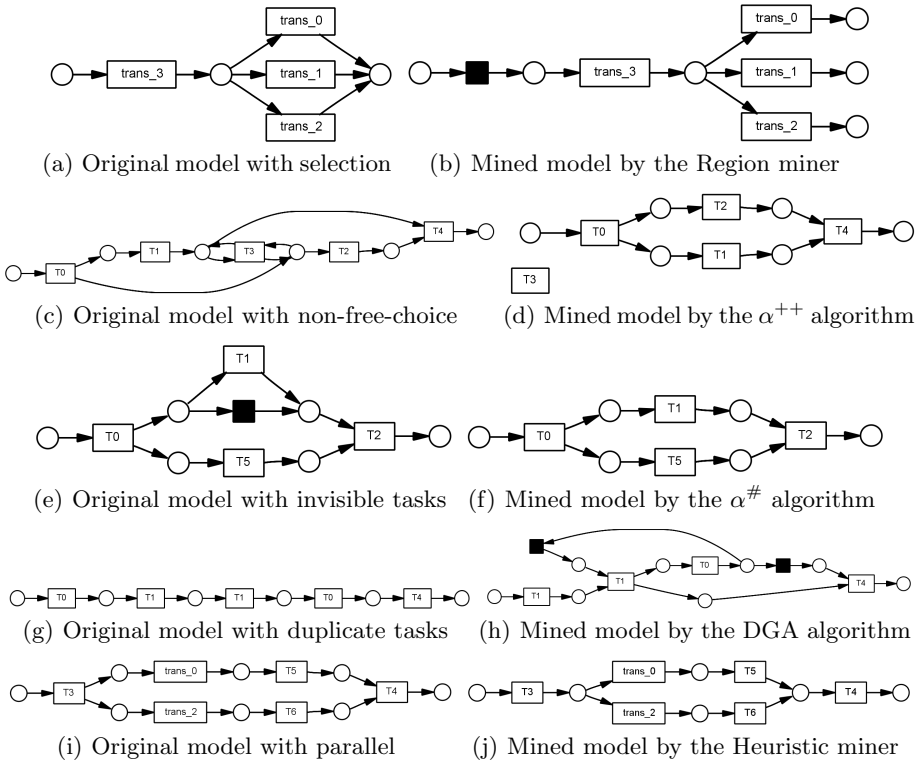
Table 6. Evaluation results based on 70 artificial process models

Category	Similarity	α	$\alpha^{\#}$	α^{++}	DGA	GA	Heu	Reg
seq	TAR	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	CON	1.00	1.00	1.00	1.00	1.00	1.00	0.52
par	TAR	1.00	1.00	1.00	0.94	0.97	0.58	1.00
	CON	1.00	1.00	1.00	0.67	0.95	0.14	0.65
sel	TAR	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	CON	1.00	1.00	1.00	0.94	0.94	1.00	0.25
slp	TAR	1.00	1.00	1.00	0.99	1.00	0.92	0.77
	CON	1.00	1.00	1.00	0.24	0.24	0.34	0.37
nfc	TAR	0.81	0.92	0.93	0.91	0.97	0.64	0.91
	CON	0.55	0.43	0.86	0.16	0.29	0.19	0.34
inv	TAR	0.60	1.00	0.72	0.98	0.95	0.85	0.82
	CON	0.20	0.86	0.20	0.58	0.66	0.67	0.11
dup	TAR	0.44	0.88	0.43	0.62	0.61	0.41	0.83
	CON	0.24	0.14	0.24	0.43	0.13	0.17	0.09

There are also some anomalies perceived from the evaluation results. Firstly, the Region miner has lower structure similarities almost all the time. This can be demonstrated in Figure 2(a) and Figure 2(b). Compared to the original model, the Region miner is customized to add a manual invisible task at the beginning of the mined model and construct several sink places at the end of it in case of OR-join. Secondly, the α^{++} algorithm does not get TAR=1 and CON=1 on non-free-choice constructs. This is because two reference models in this category are not minded correctly, one of which is shown in Figure 2(c) as well as the mined model in Figure 2(d). $T3$ is involved in a short loop and non-fee-choice construct in the original model. However, it is isolated in the mined model. Thirdly, the $\alpha^{\#}$ algorithm does not get CON=1 on invisible tasks. The reason can be explained by Figure 2(e) and Figure 2(f). The $\alpha^{\#}$ algorithm cannot rediscover the invisible task involved in just one of multiple parallel branches which connects the AND-split transition and the AND-join transition. Fourthly, the DGA algorithm get relatively low TAR and CON on duplicate tasks although it is still the best one. See Figure 2(g) and Figure 2(h), maybe for parameters that are not optimal and log is not enough, the DGA algorithm constructs a far different mined model compared to the original one. Fifthly, the Heuristic miner get very low TAR and CON on parallel. It often rediscovers a selection structure for a parallel structure because of low frequencies, shown in Figure 2(i) and Figure 2(j).

4.2 Empirical study on real-life models

Considering non-free-choice not existing in the real-life process model repository, we use the evaluation system to evaluate the 6 process mining algorithms, excluding the α^{++} algorithm. The benchmark results are listed in Table 7.



(a) Original model with selection (b) Mined model by the Region miner

(c) Original model with non-free-choice (d) Mined model by the α^{++} algorithm

(e) Original model with invisible tasks (f) Mined model by the $\alpha^{\#}$ algorithm

(g) Original model with duplicate tasks (h) Mined model by the DGA algorithm

(i) Original model with parallel (j) Mined model by the Heuristic miner

Fig. 2. Some problematic models for different mining algorithms

From Table 7, a conclusion can be drawn that the $\alpha^{\#}$ algorithm has the best performance on all the categories. By contracting each pair of the original model and the mined model by the $\alpha^{\#}$ algorithm visually one by one, we find that the $\alpha^{\#}$ algorithm always tries to construct a understandable and sound WF-net with minimal invisible tasks. To investigate the problems encountered by other mining algorithms, deep analysis has been done on three representative models.

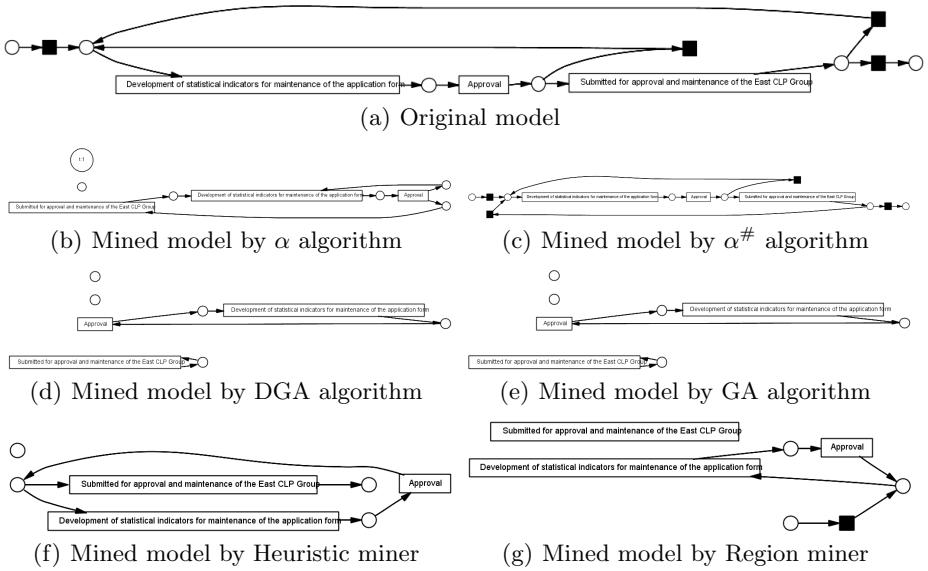
The model CO004, the maintenance process for modification of the master data of statistical cost indicators in the CO module, has two nested loops with four invisible tasks. From Figure 3, we can see that the $\alpha^{\#}$ algorithm rediscovers a model with equivalent structure to the original model and the other 6 algorithms encounter different problems, which are illustrated below. The left 5 algorithms all return isolated nets with just one loop.

The model FA306, a provision process for impairment of assets in the FA module, has one big loop with three invisible tasks and ends with two parallel branches. It is not a WF-net but sound enough. From Figure 4, we can see that

Table 7. Evaluation results based on real-life process models

Category	Similarity	α	$\alpha^{\#}$	DGA	GA	Heu	Reg
dup	TAR	0.40	0.55	0.55	0.54	0.38	0.37
	CON	0.10	0.16	0.06	0.04	0.13	0.10
inv	TAR	0.72	0.90	0.74	0.73	0.66	0.75
	CON	0.38	0.59	0.39	0.40	0.41	0.22
slp	TAR	0.06	1.00	0.40	0.40	0.25	0.48
	CON	0.06	0.90	0.19	0.18	0.19	0.12
nlp	TAR	0.00	1.00	0.25	0.25	0.00	0.57
	CON	0.04	1.00	0.05	0.05	0.05	0.11
nwf	TAR	0.77	1.00	0.89	0.90	0.86	0.94
	CON	0.37	0.47	0.38	0.41	0.40	0.38

the $\alpha^{\#}$ algorithm rediscovers a sound WF-net with equivalent behaviors but more compact and clear structure compared to the original model. The other 6 algorithms encounter different problems, which are illustrated below. The α algorithm constructs an isolated net with deadlocks. The mined model by the Region miner is relatively good except that it ends with two parallel branches and its behaviors are not equivalent to those of the original model. The other 3 algorithms rediscover isolated nets with one or two loops.



(a) Original model

(b) Mined model by α algorithm

(c) Mined model by $\alpha^{\#}$ algorithm

(d) Mined model by DGA algorithm

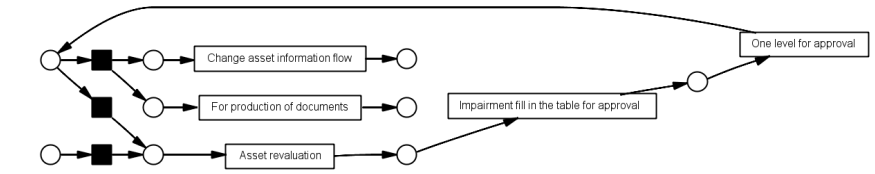
(e) Mined model by GA algorithm

(f) Mined model by Heuristic miner

(g) Mined model by Region miner

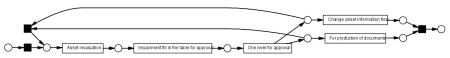
Fig. 3. Case study: CO004

The model FA404, the process of the year-end accounting checkout in the FA module, has one short loop with two invisible tasks and two duplicate tasks, mismatches AND-Split with XOR-Join. It is not a sound WF-net. From Figure 5, we can see that the $\alpha^{\#}$ algorithm rediscovers a sound WF-net with equivalent behaviors but more compact and clear structure. The $\alpha^{\#}$ algorithm imitates the behaviors of duplicate tasks by invisible tasks involved in a loop. It also constructs an AND-Join invisible task to match the previous AND-Split. The other

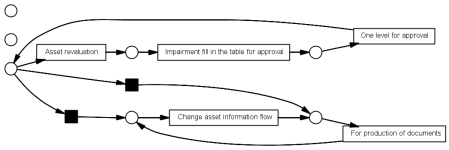


(a) Original model

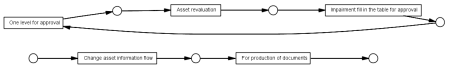
(b) Mined model by α algorithm



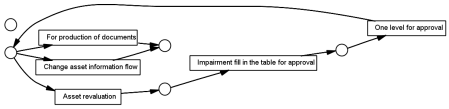
(c) Mined model by $\alpha^{\#}$ algorithm



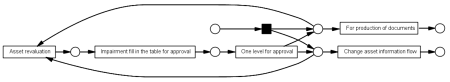
(d) Mined model by DGA algorithm



(e) Mined model by GA algorithm



(f) Mined model by Heuristic miner



(g) Mined model by Region miner

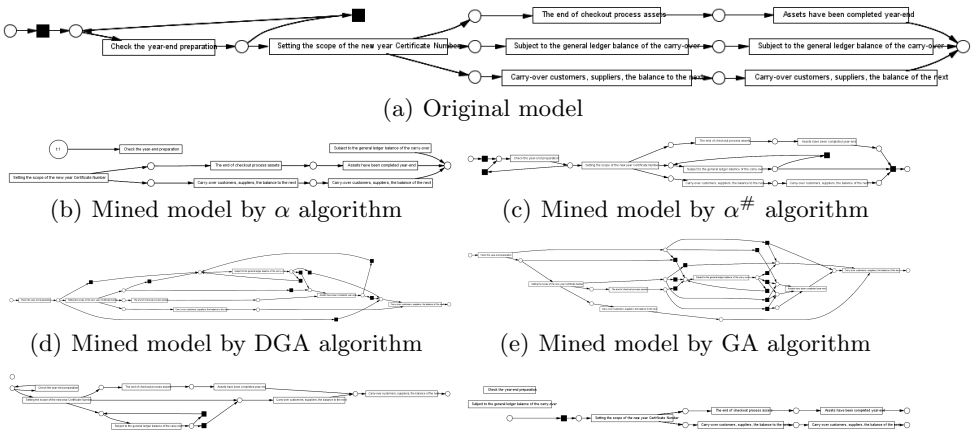
Fig. 4. Case study: FA306

6 algorithms encounter different problems, which are illustrated below. The α algorithm constructs isolated nets still with XOR-Join. The GA algorithm returns a complex WF-net with too many invisible tasks and the DGA algorithm mines a similar model with more compact structure. The Heuristic miner rediscovers an isolated net with two short loops. The mined model by the Region miner is isolated with no loops and ends with parallel branches.

In the above two experiments, the GA, DGA, Heu, Reg algorithms did not present as good performance as we expected. By investigating the generated logs thoroughly, we found that although they are complete according to TAR completeness, the requirements on log completeness of GA, DGA, Heu, Reg algorithms are far from satisfied. GA, DGA and Heu algorithms consider frequencies of events and traces, while Reg algorithm need global completeness. Furthermore, we specify a big number for traces in the generated logs by adjusting the parameter of the log generator and do the testing again. The performance of these four algorithms are improved gradually.

5 Conclusion and Future Work

We can draw the following conclusions from our empirical study. (1) The event logs and reference models are all valid inputs for evaluating the process mining algorithms. As we observed, organization users are more familiar with the process models in their laptop computers and are less familiar with the execution



(a) Original model

(b) Mined model by α algorithm

(c) Mined model by $\alpha^{\#}$ algorithm

(d) Mined model by DGA algorithm

(e) Mined model by GA algorithm

(f) Mined model by Heuristic miner

(g) Mined model by Region miner

Fig. 5. Case study: FA404

logs in the back-end servers. The evaluation results comparing with the reference models are more clear and understandable to the end users. (2) We can find a proper process mining algorithm which can satisfy an enterprise's requirements, e.g. the $\alpha^{\#}$ algorithm [16] is an ideal process mining algorithm for the boiler manufacturer. (3) We can find the problematic phenomenon with the real-life reference model easily, and are more helpful to improve the process mining algorithm by process researchers. (4) It is very interesting that we can use the process mining algorithm to improve some real-life models, such as reducing the dead tasks, the unnecessary tasks, the none-sound structures etc..

Comparing with the goal of constructing a widely-accepted benchmark, our empirical study is very primary. The following issues should be addressed in future work. Firstly, the log quality metric, such as ***completeness***, should be developed and a controllable log generator should be designed. Secondly, optimal methods for parameter setting should be investigated for some mining algorithms, e.g., the genetic algorithm. Thirdly, the association between the reference model and its generated event logs should be considered together.

References

1. R. Agrawal, D. Gunopulos, and F. Leymann. Mining process models from workflow logs. In I. Ramos, G. Alonso, and H.J. Schek, editors, *Sixth International Conference on Extending Database Technology*, pages 469–483, 1998.
2. A.K.A. de Medeiros, B.F. van Dongen, W.M.P. van der Aalst, and A.J.M.M. Weijters. Process mining: extending the α-algorithm to mine short loops. BETA Working Paper Series, WP 113, Eindhoven University of Technology, 2004.
3. A.K.A. de Medeiros, A.J.M.M. Weijters, and W.M.P. van der Aalst. Genetic process mining: an experimental evaluation. *Data & Knowledge Engineering*, 14:245–304, 2007.
4. J. Herbst and D. Karagiannis. Workflow mining with InWoLvE. *Computers in Industry*, 53(3):245–264, 2004.

5. J. Li, D. Liu, and B. Yang. Process mining: An extended -algorithm to discovery duplicate tasks. *CHINESE JOURNAL OF COMPUTERS*, 30(8):106–110, 2007.
6. Jan Mendling. *Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness*, volume 6 of *Lecture Notes in Business Information Processing*. Springer, 2008.
7. A. Rozinat, A.K.A. de Medeiros, A.J.M.M. Weijters, and W.M.P. van der Aalst. The need for a process mining evaluation framework in research and practice. In A. ter Hofstede et al., editor, *BPM 2007 Workshops*, volume 4928 of *Lecture Notes in Computer Science*, pages 84–89. Springer-Verlag, Berlin, 2008.
8. A. Rozinat and W.M.P. van der Aalst. Conformance checking of processes based on monitoring real behavior. *Information Systems*, 33(1):64–95, 2008.
9. G. Schimm. Process miner - a tool for mining process schemes from event-based data. In S. Flesca and G. Ianni, editors, *JELIA 2002*, volume 2424 of *Lecture Notes in Artificial Intelligence*, pages 525–528. Springer-Verlag, Berlin, 2002.
10. W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow mining: discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.
11. B.F. van Dongen, A.K.A. de Medeiros, H.M.W. Verbeek, A.J.M.M. Weijters, and W.M.P. van der Aalst. The prom framework: a new era in process mining tool support. In G. Ciardo and P. Darondeau, editors, *ICATPN 2005*, volume 3536 of *Lecture Notes in Computer Science*, pages 444–454. Springer-Verlag, Berlin, 2005.
12. B.F. van Dongen and W.M.P. van der Aalst. EMiT: a process mining tool. In J. Cortadella and W. Reisig, editors, *ICATPN 2004*, volume 3099 of *Lecture Notes in Computer Science*, pages 454–463. Springer-Verlag, Berlin, 2004.
13. A.J.M.M. Weijters and W.M.P. van der Aalst. Rediscovering workflow models from event-based data using little Thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162, 2003.
14. A.J.M.M. Weijters, W.M.P. van der Aalst, and A.K.A. de Medeiros. Process mining with heuristicsminer algorithm. In *BETA Working Paper Series, WP 166, Eindhoven University of Technology, Eindhoven*, 2006.
15. L. Wen, W.M.P. van der Aalst, J. Wang, and J. Sun. Mining process models with non-free-choice constructs. *Data Mining and Knowledge Discovery*, 15(2):145–180, 2007.
16. L. Wen, J. Wang, W.M.P. van der Aalst, B. Huang, and J. Sun. Mining process models with prime invisible tasks. *Data & Knowledge Engineering*, 69(10):999–1021, 2010.
17. H. Zha, J. Wang, L. Wen, C. Wang, and J. Sun. A workflow net similarity measure based on transition adjacency relations. *Computers in Industry*, 61(5):463–471, 2010.