A Probabilistic Approach for Process Mining in Incomplete and Noisy Logs

Roya Zareh Farkhady and Seyyed Hasan Aali

***Abstract*— Process mining techniques aim at automatically generating process models from event logs. Many existing techniques in process mining are applicable only under very restrictive conditions about the log completeness. However, due to the completion time of the tasks in the model, the generated logs in most of the real life processes do not satisfy these conditions. In many real life logs, too many log instances are needed for the mining approach to work properly. Therefore, another definitions, metrics and algorithms are required to mine event logs with not enough instances. In this paper, a probabilistic approach is proposed to mine event logs when the number of instances is limited. In order to evaluate the proposed approach, time Petri nets are used to generate more realistic event logs. In comparison with many existing approaches, based are the results of our experiments, the proposed approach is very robust in mining process logs with high degrees of parallelism, incompleteness and noise.**

***Index Terms*— process mining, incomplete, noisy, log, probabilistic**

I. Introduction

Nowadays, information systems are widely used to support the execution of business processes. These information systems typically support logging capabilities that register what has been executed in the organization. These produced logs, called event logs, usually contain data about how people and procedures in an organization work [1]. Process mining techniques are used to discover useful information from the event log, the extracted information can be used to deploy new systems that support the execution of business processes, to find out how to analyze and improve the already enacted process. In systems with no explicit model of the underlying process, process mining techniques can be used to discover the process model and even if an existing process model is expected to be executed these techniques help to find out what is really happening in the organization which may differ considerable form what is assumed to happen [2].

Four different perspectives could be distinguished in process mining; control-flow perspective, the organization perspective, the information perspective, and the application Perspective[3]. However the dominant perspective is the so-called control-flow perspective which mines a model of the process, which specifies the ordering relations between tasks in an event log. In order to mine the control-flow perspective, the workflow log should contain a set of records such that: each record is about an event referring to a task in the workflow instance, it is possible to infer the order in which the tasks are executed. Events also may have timestamps representing the time at which the task is done or recorded. These timestamps are used in some process mining works to adding time information to the process model or to improve the quality of the discovered process model [2].

There are some challenging problems in mining an event log [3], [4] three of the most challenging problems are about mining noisy and incomplete logs and also imbalance execution priorities. Real life logs are almost noisy and incomplete. Therefore in order for a mining algorithm to be applicable it should be able to distinguish exceptions from the normal flow of the model, heuristic approaches are proposed to deal with noise and incompleteness, in these approaches to protect the induction process against inferences based on noise, only task-patron-occurrences above a threshold frequency are assumed reliable enough for the induction process [5].

logs with high amounts of noise they cannot distinguish among the rare cases mentioned above and the noisy log instances.

Since the issues of noise, imbalance and incompleteness are related, the problem discussed above can be considered to be caused by log incompleteness. A log is said to be incomplete if it does not contain sufficient information to derive the process [3]. However different approaches may need different amount of data to extract the correct model out a single workflow log, therefore an approach is considered to be more robust with respect to noise if it needs fewer log instances to extract the correct model.

In some of the existing works, in order to see the impact of noise, imbalance and incompleteness on the behavior of a mining algorithm, the benchmark event logs are created via simulation, in these works, CPN-Tools [8] are employed to simulate the process execution, and in order to introduce noise into the event log, some noise introducing operations are performed, however the generated event log may not be realistic since there is no time associated with the workflow tasks, However in a real life situation each well-defined task in a workflow model needs some time to be completed,

Manuscript received January 4, 2011; revised January 17, 2011.

Roya Zareh Farkhady Author is with the Computer Engineering Department, Islamic Azad University Of khodafarin, Iran, (e-mail: roya_farkhady@yahoo.com).

Seyyed Hasan Aali Author is with the Computer Engineering Department, Islamic Azad University Of khodafarin, Iran, (e-mail: seyyed.hasan.aali@gmail.com)

and when different tasks in the model have different distributions for their completion times, in the generated workflow model some task-sequences may be more frequent while some others are very rare. Therefore assigning time with workflow tasks causes the generated workflow log to be more similar to the real life logs.

In this paper a probabilistic approach is presented to deal with the issues of noise, imbalance and incompleteness, in order to remove some of the restrictions in the current approaches we introduce our metrics to decide about the basic dependencies among the tasks. We use the time Petri nets instead of CPN-Tools to generate more realistic workflow logs with different amounts of imbalance and incompleteness; also we use some noise introducing operations similar to the operations used in [5] to generate event logs with different amounts of noise. Our experimental evaluations show that the proposed approach in capable of achieving more accurate results than the many of the present approaches in dealing with noisy and incomplete logs.

II. RELATED WORKS

The first papers on workflow mining were in the context of software engineering processes. Cook et al. [9], [10], [11], [12], [13] were the first ones to work on process mining, they used three algorithms, Rnet, which was a purely statistical approach, KTail which was a purely algorithmic approach and a Monrovian approach which was a mixture of algorithmic and statistical methods. The Monrovian approach is able to deal with noise and proved to be superior to the other two algorithms. Agrawal et al. [14] were the first ones to apply process mining in a business setting. Their algorithm is able to deal with noise and assumes that each task appears only once in a process instance. However, Herbst et al. proposed an approach [15], [16], [[17] which is able to tackle duplicate tasks. In their works a two-step approach is employed to mine the process models, in the first step the dependencies between the tasks are captured and represented by Stochastic Activity Graph (SAG). In the second step the SAG is converted to a block-structured process model represented by Adonis Definition Language. The SAG models the behavior in the log but does not contain any duplicate tasks. In order to deal with duplicate tasks, the algorithm applies a set of split operations to nodes in the SAG.

Van der Aalst et al. [7], [18], [19], [20], [21], [22] have focused on mining process models. In [13] they developed the α-algorithm and also proved to which class of models their approach is guaranteed to work. In their approach the event log is assumed to be noise-free and complete with respect to a defined notion of log completeness. However, the proposed approach had problems in mining some common constructs in workflow models. Among these constructs are short loops, which are loops of length one and two. For instance, the α-algorithm was proven to mine sound Structured Workflow nets without short loops.

The α-algorithm does not take into account the frequency of a relation. Therefore, the algorithm is very sensitive to noise and even one erroneous example can completely mess up the derivation of a right conclusion about the binary relations. In addition, the definition of completeness given in this approaches is very arbitrary and strong.

Weijters et al. [23] presented a mining algorithm that uses the ideas behind the α-algorithm and the Cook et al.'s approach. This algorithm uses the frequency of binary relations among tasks to infer the basic relations and is able to mine noisy and incomplete logs. The main idea behind the heuristics is that the more often task A follows task B and the less often B follows A, the higher the probability that A is a cause for B. The algorithm is implemented as the Heuristics miner plug-in in the ProM framework tool [22].

Most of the existing algorithms in workflow mining are not able to find the long distance dependencies among tasks. Due to the local strategy used in some algorithms only the local constructs can be mined. However, Wen et al. [23], [21] have proposed two extensions of the α-algorithm, one of these extensions, the β-algorithm [18], is based on the assumption that the tasks in the log are non-atomic. The other extension, the α^{++}-algorithm [22], uses the non-local information in the log to mime Petri nets with local or non-local non-free choice constructs.

III. PRELIMINARIES

The notions of workflow log and workflow log instance are defines as follows:

Definition 1. (Workflow log instance, workflow log)

- Let $T = \{t_1, t_2, ..., t_n\}$ be a set of tasks, every non-empty sequence over T is called a **workflow instance**.
- A **workflow log** is a bag of workflow instances.

We also define some basic relations on the set of tasks, T, as follows:

Definition 2. The basic relations are defined as follows:

$\forall$ A, B (A$\neq$B) in T: A $\|$ B if and only if A and B are concurrent and $A \rightarrow B$ if and only if A is a direct successor of B and B is not a direct successor of A. $A \rightarrow A$ if and only if there is a length-one loop [21] in A. $A \rightarrow^{-1} B$ if and only if there is a length-two loop between A and B. A # B if and only if not (A $\|$ B or $A \rightarrow B$ or $B \rightarrow A$ or $A \rightarrow^{-1} B$).

Finding these basic relations, direct successors are determined and knowing the direct successors, the Petri net model of the underlying process could be constructed. However, this paper does not discuss how the model could be constructed out of the direct successor relation. Fortunately, this process is rather mechanical and is presented in [5]. In cases where a representative and a sufficient large subset of possible behaviors of the process are registered in the log, and the log contains no noise, it is quite simple to determine the basic relations. However, having a workflow log with no enough instances or a log with too much noise it could be a very challenging work to

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

decide if between two events say a and b each of the basic relations holds.

In order to find the basic relations, some information is abstracted out of the workflow log; the proposed algorithm and algorithm presented in some of the existing approaches use the following information:

For each task A, #A denotes the overall frequency of task A in the workflow log, and for each two (different or same) tasks A and B, the following information is used: # AB: the frequency of A directly preceded by B. # A..B: the frequency of A directly or indirectly followed by task B but before the next appearance of A. # A..B..A: the frequency of A directly or indirectly followed by task B but before the next appearance of A and then B directly or indirectly followed by task A but before the next appearance of B, and # A..B..B: the frequency of A directly or indirectly followed by task B but before the next appearance of A and then B directly or indirectly followed by task B but before the next appearance of A.

The above mentioned information is used in our algorithm and the algorithms presented in the prior approaches to find the concurrency, causality, different types of loops.

IV. The Proposed approach

In this section we present our approach in finding the direct successors. There are some important problems a process mining approach should overcome; dealing with noise, imbalance, incompleteness and mining event logs with high degrees of parallelism. In order to deal with this challenges new concepts and metrics need to be introduced. Since knowing the direct successors the Petri net model can be constructed out, therefore determining the direct successors is essential in mining a workflow model, therefore some metrics and concepts are needed to decide about the direct succession relation. One of this metrics is the mean distance metric discussed in the following.

A. Motivation

Some motivating issues are discussed in this section, showing the usefulness of our approach. In [6] a metric is used to indicate how certain we are that the (A, B) truly belongs to the $\rightarrow$ relation, the notation is $A \Rightarrow_W B$ and is calculated as follows:

$$A \Rightarrow_W B = \left(\frac{\#AB - \#BA}{\#AB + \#BA + 1} \right)$$

The values of $\Rightarrow_W$ between the events are used to determine the true direct successors, if the value of $A \Rightarrow_W B$ is above a certain threshold, say p, then it is induced that B is a direct successor of A. Although for many dependency relations it is unnecessary to use a threshold value and the all-activities-connected heuristic [6] helps we to take the direct successors but there are many cases in which a threshold value seems to be necessary. The threshold value should be selected with respect to the amount of noise and the degree of concurrency and imbalance in the event log. However different parts of the model may have different degrees concurrency and imbalance, therefore a single value of p cannot be chosen suitable or all parts of the model. This might lead to a wrong derivation even in the simple case in Fig. 1 When due to the completion time distributions of the tasks we have:

$$N = \#BD + \#DB,\ \#DB / N = e$$

And e is a small real number, since for large values of N we have:

$$B \Rightarrow_W D \approx \left(\frac{(1-2e)N}{N+1} \right) \approx (1-2e)$$

Therefore we have $B \Rightarrow_W D \approx (1\text{-}2e)$ and whatever the threshold value, p, is, for small values of e we might have (1-2e) > p and it is derived that D is a direct successor of B.

In [5] some other rules are proposed to decide whether the (A, B) belongs to the $\rightarrow$ relation, in that approach

a subtle approach is presented to mine noisy and incomplete logs. In their approach some metrics are used to determine the direct successors, the metric $\#A \rightarrow_W B$ is used to show the strength of the causal relation between tasks A and B and is calculated dividing the $\#A \rightarrow_W$ B-causality counter by the minimum overall frequency of task A and B. One major rule is used in this heuristic approach:

If (($\#A \rightarrow W$ B > N) and (#AB > θ) and ((#BA < θ)) then B is a direct successor of A

The value θ is automatically calculated using the following equation: θ =1+Round (N×#L/#T). Where N is the noise factor and #L is the number of workflow instances in the workflow log, and #T is the number of tasks.

In estimating the strength of the causal relation between the tasks better results achieved when the metric $\#A \rightarrow_W B$ is calculated as follows:

$$A \rightarrow_W B = \frac{A \rightarrow_W B - causality\ counter}{\#A..B}$$

Even if B is a direct successor of A, in an incomplete log there may be no instances in which B directly follows A, such a problem may also rise in models with high degree of concurrency, better criterion may be proposed to decide about causal relation when mining an incomplete log.

Much of the problems with the current approaches are caused by log incompleteness. Log incompleteness can be more serious when dealing with high degrees of concurrency. Determining the concurrent tasks in mining an incomplete event log can be a challenging issue.

B. Generating Workflow Logs

Using real-life logs seems to be the natural choice for testing the process mining algorithms. However, in order to tune and test a mining algorithm a researcher need to have more control about the properties of the event log [7].. Therefore, the advantages and merits of new mining algorithms can be best revealed in mining logs which are created via simulation. In [7] CPN tools are extended to generate event logs. However, we believe that the created logs will be more realistic if time Petri nets are employed instead of CP-nets. In a real life situation, when the preconditions of a task become true it takes some time for the task to be done and recorded. The task completion time could have any distribution.

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

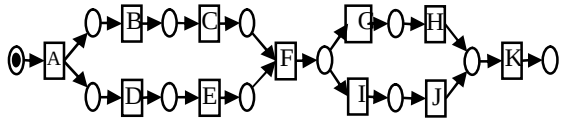

Fig. 1. An example of a workflow net

As an example consider the workflow model shown in Fig. 1, if a CPN is used for log generation, in about half of the log instances the task B precedes D and in the other log instances D precedes B. However, this is not the case in an environment where the tasks B and D each have a completion time distribution. For example if the completion time for the task B, TB, has a normal distribution with μ = 10 and $\sigma 2$ = 4 and TD has a deterministic value 13, in about 93% of the log instances B precedes D. Now suppose that TC has a deterministic value 9, and TE has an exponential distribution with parameter λE = 3, with these completion time distributions for the Tasks B, C, D and E, in about 17% of the instances C precedes E. Therefore, when an event log is generated via simulation of a CP-net, the generated workflow log is not similar to a real life log. However, mining the processes generating workflow logs with such behaviors is more challenging than extracting the process model out of a workflow log that is created via simulation of a CP-net.

When generating event logs using the CPN-Tools, no completion time is associated with the tasks and the next task to be recorded is chosen non-deterministically from the ongoing tasks. It is as we assume that all the tasks in the model have the same completion time distributions, say a normal distribution. However, this is not a realistic assumption; assigning completion time with tasks will cause the event log to be more realistic. In a real life environments where logs are generated each task is consisting of some subtasks each of which has a completion time. The completion time of the whole task is the summation of these values.

C. Introducing noise

Some mining algorithms assume all the information in the event log to be correct. However, in most situations this is not the case, the log may contain noise, incorrectly logged information. Like the method used in [5] we incorporate noise by performing some noise introducing operations, such as; Deleting the head of the event sequence, Deleting the tail of the event sequence, Deleting a part of the body of the event sequences, Interchanging two randomly chosen events, Shifting a randomly chosen event to the right, and Shifting a randomly chosen event to the left.

D. Mean distance metric

In this section we introduce a new metric used to decide about the direct succession relation.

We assume (A, B) to belong to the direct succession relation if there are enough instances in which B appears shortly after A. When B is a true direct successor of B, it is likely to have enough instances in the log to show this behavior. If there is no task concurrent to A or B then we expect that B appears directly after A and the distance between A and B in the log is expected to be 1. However if there are tasks concurrent to A or B, the tasks may be appeared between A and B, in the log instances, and if all the occurrences of these concurrent tasks are removed from the log in the resulted log, B appears directly after A and the distance between A and B is resulted to be 1.

Consider again the workflow model shown in Fig. 1. Although the topple (B, C) belongs to the direct succession relation, but if task C requires a long completion time, there may be no instance in which B is directly followed by C. Assume there are 100 log instances containing both B and C, in 56 instances D, in 35 instances E, in 9 instances DE appears between the tasks B and C. Also assume that we know:

$$\{(C,H),(D,H),(E,H),(C,I),(D,I),(E,I)\} \subset \|$$

By removing all the concurrent tasks D and E, 100 cases will be resulted in which the distance between B and C is 1. This helps us to decide that C is a direct successor of B. However, we are never sure if two tasks are in parallel, we only have some probabilities about concurrency between tasks. These probabilities are obtained based on the observations. The expected value for the distance between T_i and T_j is denoted by $\delta(T_i, T_j)$ and calculated as follows:

$\delta(Ti, Tj) = 1$;
  for each T_k between Ti and Tj
    if DS(Ti, Tk) and DS(Ti , Tj) then $\delta(Ti, Tj) = \delta(Ti, Tj) + (1-P(Tk \| Tj))$
    else if DS(Tk, Tj) and DS(Ti , Tj) then $\delta(Ti, Tj) = \delta(Ti, Tj) + (1-P(Ti \| Tk))$
    else $\delta(Ti, Tj) = \delta(Ti, Tj) + (1-(P(Ti \| Tk) \times P(Tk \| Tj)))$
  if $\delta(Ti, Tj) > 3$ then $\delta(Ti, Tj) = 3$

Where DS(Ti, Tk) is calculated as follows:

$N = \#AB + \#BA$

$M = \#ABA + \#BAB$

$F(A \rightarrow B) = (\#AB - \#BA)/(N+1)$

$F(A \| B) = (1 - |F(A \rightarrow B)|) \times N/(N+1)$

$F(A \# B) = 1 - N/(N+1)$

$if\ (M > 5\theta)\ then\ F(A \square B) = 1\ else\ F(A \square B) = 0$

$if\ (F(A \rightarrow B) > F(B \rightarrow A)\ and\ F(A \rightarrow B) > F(A \| B)\ and\ F(A \rightarrow B) > F(A \# B)\ and\ F(A \rightarrow B) > F(A \square B)\ then\ DS(A,B) = true\ else\ DS(A,B) = false$

Where $F(A \rightarrow B)$, $F(A \| B)$, $F(A \# B)$ and $F(A \square B)$ indicate how certain we are that the topple (A, B) belongs to the $\rightarrow$, $\|$, #, relation respectively. We set DS(A, B) = true if according to this metrics $F(A \rightarrow B)$ has a larger value than the other three metrics.

When in a workflow model the task B is a direct successor of A and in a log instance C is appeared between A and B then it can be inducted there can be three possibilities; C is a successor of A and is parallel to to B, C is a predecessor of B and is parallel to A, C is parallel to both the tasks A and B. the first possibility shows the motivation for the first line in the body of the for-loop. The task A in this model is an AND-SPLIT. Since the tasks B and C are in parallel, in calculating $\delta(A, B)$, the distance A and B, for instances containing the substring ACB, the task C should not be removed. Therefore, if we are pretty sure that both the topples (A, B) and (A, C) belong to the $\rightarrow$ relation, then we have:

$$\delta(A, B) = 1 + (1 - P(Tk \| Tj))$$

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

A similar reasoning about third possibility can illustrate motivation for the third line in the for-loop body, the motivation behind the second line in the body of the for-loop is explained using the part of workflow model in second possibility. Now consider the distance between A and C, in every log instance B resides between A and C, and since P(A || B) and P(C || B) are both small numbers near to 0, δ(A, C) is about 2.0. Since there may be different path between A and B, therefore we need to see if there are enough cases in which we the distance between A and B is less than 1.1, so we use the notation $\overline{\delta}(A,B)$ to denote the mean distance.

E. Dealing with Concurrency

In order to calculate the distance between two tasks we need to decide if two tasks are concurrent, however, we don't know for sure whether two tasks are in parallel, we may just have an estimation of the probability of A and B be in parallel, we tried to propose a metric how sure we are that two tasks, say A and B, are concurrent. Our estimation of this probability is as follows:

$$p(A\|B)=\frac{N}{N+1}e^{-((1.25/(0.5-\theta))(x-0.5))^K},\ x=\frac{\#A.B}{N},\ N=\#A.B+\#B.A$$

The tasks A and B are said to be concurrent if they can be executed in any order, in cases where the event log is assumed to be complete and noise free it is very easy to determine concurrent tasks, in such a situation tasks A and B are determined to be concurrent if (#AB>0 and #BA>0). In noisy situations decisions may be made based on the frequencies and only task-patron-occurrences above a threshold frequency are reliable enough for our induction process, however when mining an incomplete log, deciding based on the frequencies may lead to incorrect results. In our approach, if both the #A..B and #B..A are above a certain threshold we may conclude that A and B are concurrent; however this is an efficient condition and is not necessary as shown before. The above formula is used to indicate how sure we are that two task are truly concurrent: Where N is summation of #A..B and #B..A and x is the fraction of times A appears before B and θ is calculated according to the following:

$$\theta = 1 + Round(\frac{F \times L}{3T})$$

Where F is the noise factor, L is the number of workflow instances containing both the task A and B and T is the number of task types in the workflow log, K in the above formula can be any even number, in our experiments we set K = 20, the plot for K=10 and N = 80 is shown in figure 3. As one can see for x < θ and x > 1 - θ the probability of the A and B to be in parallel is estimated to be zero and only for (θ < x < 1- θ) the probability has a non-zero value.

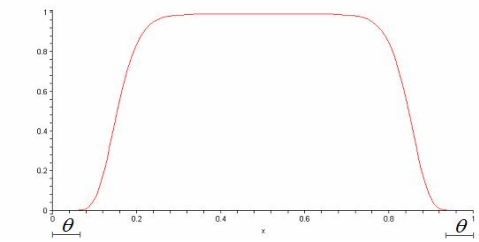

Figure 2. The probability of two tasks being in parallel as a function of #A..B and #B..A

Our estimation of this probability is based on the event log which might be incomplete.

F. outline of the Algorithm

Our algorithm in finding the direct succession relation is based on the simple metric called mean distance among tasks. Using the event log we try to calculate the distance between each two none concurrent tasks in the model, task B is assumed to be a direct successor of A if there are enough instances in which δ(A, B) is about 1.0. We use this rule to decide about the direct successors:

P(A||B)<0.2 and $\overline{\delta}(A,B)$ < 2.5 and F(Support(A→B)) >0.9

V. Experimental Results

In order to evaluate the proposed algorithm, some experiments have been done on five time Petri net models. The models have complexities comparable with the complexity of models used in [6]. The models contain self-loops, length two loops and other kinds of loops. the degrees of parallelism in the models ranges from 2 to 5 execution threads, have not duplicate tasks and no hidden task exists except that all the AND-SPLIT, OR-SPLIT, AND-JOIN and OR-JOIN task supposed to be hidden. number of tasks in this models ranges from 24 to 67 tasks.

All the models have been used with Different degrees of imbalance and different completion time distributions assigned to the tasks, in order to see the impact of noise on our mining algorithm the experiments are done with different amounts of noise in the event log, events log without noise, with 5%, 10%, 20%, 30%, 40% and 50% noise are used to show the behavior of the proposed algorithm. Also the experiments are repeated with different amounts of imbalance in the branch points.

Assigning different completion time distributions to the tasks, different workflow logs are generated and mined. The completion time assigned to the tasks in our experiments is the summation of two parts, a deterministic

TABLE 1: RESULTS OF ALGORITHM

Imbalance in branch points: 50%		Amount of noise				
		0%	5%	10%	20%	30%
Number of instances	10	0.919, **0.961**	0.926, **0.967**	0.918, **0.967**	0.914, **0.959**	0.921, **0.962**
	20	0.887, **0.961**	0.886, **0.966**	0.893, **0.966**	0.902, **0.967**	0.918, **0.968**
	30	0.873, **0.960**	0.882, **0.962**	0.883, **0.963**	0.887, **0.968**	0.912, **0.965**
	50	0.872, **0.962**	0.861, **0.965**	0.868, **0.962**	0.884, **0.961**	0.917, **0.960**
	100	0.868, **0.961**	0.875, **0.959**	0.876, **0.957**	0.884, **0.956**	0.914, **0.953**
	200	0.883, **0.954**	0.892, **0.963**	0.880, **0.960**	0.907, **0.952**	0.928, **0.951**
	300	0.916, **0.955**	0.915, **0.957**	0.908, **0.960**	0.919, **0.960**	0.946, **0.952**
	500	0.940, **0.967**	0.937, **0.961**	0.931, **0.965**	0.936, **0.962**	0.953, **0.959**
	1000	0.976, **0.975**	0.962, **0.970**	0.973, **0.977**	0.974, **0.975**	0.963, **0.969**
	2000	0.988, **0.988**	0.979, **0.979**	0.976, **0.981**	0.980, **0.981**	0.972, **0.974**

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

random variable randomly chosen in the interval (10, 15) and a normal distribution random variable with $\mu = 25$ and $\sigma^2 = 7$, the results from 100 executions of our mining algorithm with different completion times assigned to the tasks is shown in TABLE 1

The result of our experiments for different amounts of noise and different amounts of workflow log instances are shown in the TABLE 1, in each experiments the number of correctly determined elements in the Boolean matrix of direct successors relations is divided by the number of all the elements, the results obtained by applying the approach proposed in this paper in compared with the result of the approach presented in [5], the results of our approach are shown bold. As one As shown in fig 2.

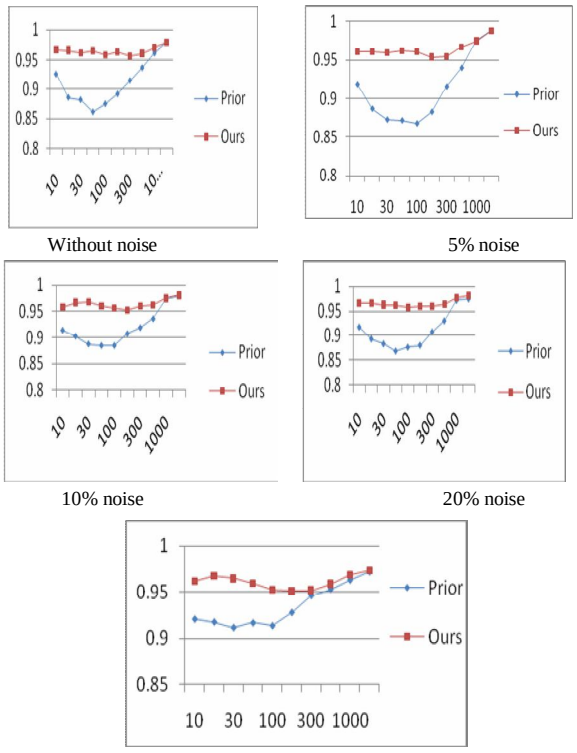

Figure 2. Evaluation results

VI. Conclusion

In this paper we have introduced a probabilistic approach for finding the direct successors in incomplete workflow logs. The proposed approach uses a two phase algorithm to find the direct successors. New metrics and concepts are introduced to use the existing event log more efficiently.

We have introduced time Petri nets, instead of CP-nets in similar works, to create more realistic logs. We have also shown that our approach is more successful than many of the existing techniques. Our experiments on 250 different workflow models shown that our approach is more robust in mining incomplete and noisy event logs.

References

[1] W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, A.J.M.M. Weijters, “Workflow mining: a survey of issues and approaches”, Data and Knowledge Engineering, 2003.

[2] W.Aalst,. van der, & Weijters, A. “Process mining: A research agenda”. Computers in Industry, 2004.

[3] A.J.M.M. Weijters W.M.P van der Aalst, “Process Mining Discovering Workflow Models from Event-Based Data”, springer, 2000.

[4] A.J.M.M. Weijters, W.M.P. van der Aalst, and A.K. Alves de Medeiros, “Process Mining with the HeuristicsMiner Algorithm”, 2006.

[5] A.K.A. de Medeiros, A.J.M.M. Weijters, and W.M.P. van der Aalst. “Genetic Process Mining: A Basic Approach and its Challenges”, Springer, 2006.

[6] W.M.P. van der Aalst and A.J.M.M. Weijters, editors, “Process Mining: Special Issue of Computers in Industry”, Elsevier Science Publishers Amsterdam, 2004.

[7] R.Silva, J.Zhang, G. Shanahan, “Probabilistic Workflow Mining”, ACM, 2005.

[8] J. Herbst and K. Karagiannis. “Workflow mining with InWoLvE”. Computers and Industry, 2004.

[9] R. Bergenthum, J.Desel, R.Lorenz, and S. Mauser, “Process Mining Based on Regions of Languages”, Proceedings of the 5th International Conference on Business Process Management, 2007.

[10] B. van Dongen, N. Busi, G. Pinna, and W. van der Aalst. “An iterative algorithm for applying the theory of regions in process mining”. Technical Report Beta rapport, 2007.

[11] R.Bergenthum, R. Lorenz, S.Mauser,” Towards Applicability of Language Based Synthesis for Process Mining”, Proceedings of the 5th International Conference on Business Process Management, 2007.

[12] W. M. P. Aalst, C. W. GÄunther, “Finding Structure in Unstructured Processes: The Case for Process Mining.”, In: Proceedings of the 7th International Conference on Application of Concurrency to System Design (ACSD), 2007.

[13] R. Bergenthum, J.Desel, R.Lorenz, and S. Mauser, “Experimental Results on Process Mining Based on Regions of Languages”, ICATPN, LNCS, 2008.

[14] J. van der Werf, B. van Dongen, C. Hurkens, and A. Serebrenik, “Process discovery using integer linear programming”, ICATPN, LNCS, 2008.

[15] Process mining group eindhoven technical university: Prom-homepage. http://is.tm.tue.nl/ cgunther/dev/prom/.

[16] Z.Huang, A.Kumar, “A Study of Process Mining: Quality and Accuracy Tradeoffs”, Proceedings of the 4th Workshop on Business Process Intelligence, 2007.

[17] W. M. P. van der, H. A. Reijers, A. J. M. M. Weijters, B. F. van Dongen, A. K. A. de Medeiros, M.Song and H. M. W. Verbeek, "Business process mining: An industrial application", Information Systems, 2007.

[18] L. Măruşter, A. J. M. M. T. Weijters, W. M. P. van der Aalst and A. van den Bosch, "A rule-based approach for process discovery: Dealing with noise and imbalance in process logs", Data Mining and Knowledge Discovery”, 2006.

[19] A. K. A. de Medeiros, C. W. Günther, A. J. M. M. Weijters and W. M. P. van der Aalst, "The need for a process mining evaluation framework in research and practice", Proceedings of the 3rd Workshop on Business Process Intelligence, 2007.

[20] B.F. van Dongen and W.M.P. van der Aalst.” EMiT: A process mining tool”. International Conference on Applications and Theory of Petri Nets, Springer-Verlag, 2004

[21] J. Herbst and D. Karagiannis. “Workow mining with inwolve”. Computers in Industry, 2004.

[22] A.K. Alves de Medeiros. “Genetic Process Mining”. PhD thesis, Eindhoven, University of Technology, Eindhoven, The Netherlands, 2006.

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)