Article ID: 1002-1175(2009)01-0107-07

A clustering based method to solve duplicate tasks problem *

SONG Jin-Liang† LUO Tie-Jian CHEN Su LIU Wei

(*Graduate University of the Chinese Academy of Sciences*, *Beijing* 100049, *China*)

(Received 14 March 2008; Revised 14 May 2008)

Song JL, Luo TJ, Chen S, *et al*. A clustering based method to solve duplicate tasks problem. *Journal of the Graduate School of the Chinese Academy of Sciences*, 2009, 26(1): 107 ~ 113

Abstract Process mining is to discover structured process description from real execution data. It helps the discovery and design of business process, and improves the existent ones through delta analysis. One of the challenging problems in process mining is how to deal with duplicate tasks. This paper provides a duplicate tasks treatment stage before the real execution of mining algorithm, which method is well compatible with existent process mining algorithms and helps them deal with duplicate tasks. In addition, this paper designs a distance measure to transfer the difference of event context into numerical form, and take advantage of such distance to distinguish duplicate tasks through clustering technology. The method in this paper is proved by experiments on typical process model having duplicate tasks.

Key words process mining, duplicate tasks, clustering

CLC TP181

1 Introduction

Process mining is the term for the method of distilling a structured process description from a set of real executions. The goal of process mining is to extract an explicit process model from event logs, i.e., the challenge to create a process model given a log with events such that the model is consistent with the observed dynamic behavior[1]. Nowadays, most organizations use information systems to support the execution of their business processes, such as Workflow Management Systems (WMS), Customer Relationship Management (CRM) systems, Enterprise Resource Planning (ERP) systems and so on[2~4]. As a result, process mining techniques gather information about what is actually happening according to an event log of an organization, and discover information which can be used to deploy new systems that support the execution of business processes or as a feedback tool that helps in auditing, analyzing and improving already enacted business processes[5].

Although process mining can be treated as an application of machine learning technology for event log, it has eleven specific challenging problems to solve, which problems are described in Ref.[1]. Among them, the second challenging problem is mining duplicate tasks, which refers to the situation that one can have a process model with two nodes referring to the same task[1].

* supported by Ministry of Science and Technology of the People's Republic of China (2005DKA64100, 2005DKA10201)

†E-mail: songjl@mails.gucas.ac.cn

Some researchers consider duplicate tasks as one of the particular constructs of process model. The other major constructs are sequences, parallelism, choices, loops, non-free-choice, and invisible tasks[1]. Different mining model supports different constructs, but none of the mining methods could support all the constructs altogether[5]. Paper [5] also tries to finish a kind of genetic process mining method that may support all possible constructs of process model, but such genetic process mining method consumes much more computing resource compared to other process mining algorithms. Additionally, when paper [5] updates its original algorithm to support duplicate tasks, the new algorithm is more complex in internal representation, fitness measurement, genetic operators, and algorithm implementation. Usually how to improve existent algorithms to support duplicate tasks is not obvious.

According to the situation of applied environment, the relationship between events in realistic log can be comprehensive, and duplicate tasks problem is one of the major challenges for the rediscovery of effective process model. A fine solution for duplicate tasks not only pushes process mining technology a step forward on practical application, but also benefits wider process models such as Behavior Pattern Mining described in Ref. [6].

In paper [7, 8], Li described an improved process mining algorithm basing on α algorithm, and named it as α^{**}. This new algorithm could deal with duplicate tasks problem for sound WF-nets on the basis of complete workflow logs, which is a subclass of WF-nets and is defined in Ref. [9]. However, when we examine the detail of α^{**}, we find out that Li's improvement is independent of the original α algorithm. Li marks all events related to duplicate tasks at the beginning, and then invokes the original α algorithm to discover the real process model. In such process, Li chooses heuristic method to identify the duplicate tasks. Consequently, we get an interesting question to consider: is that possible to isolate the duplicate tasks recognition stage from the whole process mining procedure, and introduce more methods of artificial intelligence to help existent process mining algorithms dealing with duplicate tasks problem in larger class of WF-nets.

In this paper, we will introduce an independent duplicate tasks treatment stage before the execution of primary mining algorithm, and such stage transfers the original event log to be a clean one that without duplicate tasks problem. Furthermore, we use automatic clustering technology to separate duplicate tasks in this stage.

The idea in this paper has at least the following two advantages: First, the independent duplicate tasks treatment stage is well compatible with existent process mining algorithms, and it enhances the algorithms to support duplicate tasks construct. Second, the automatic clustering technology requires no future knowledge of expecting process model. In other words, we need to know neither which tasks are involved in duplicate tasks, nor the exact how many clusters a group of duplicate tasks should be divided into. Such feature is well consistent with typical scenarios that process mining technologies apply to.

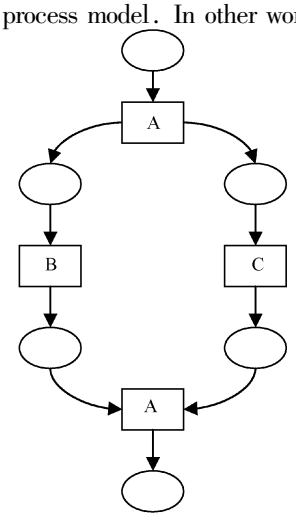

Fig.1 Example of duplicate tasks (B.33:herbstFig.5p1AND)

In structure of this paper, section 2 explains the concept of duplicate tasks problem, section 3 introduces the target and the challenges of duplicate tasks treatment, section 4 describes the detailed algorithm, section 5 presents our way of experiment design and discusses the experiment results, and section 6 concludes the paper and talks about the direction of future works.

2 What is duplicate tasks

The term duplicate tasks refers to the situation that one can have a process model with two nodes referring to the same task[1]. Fig.1 is an example of process model that contains duplicate tasks, and it is the same model as Figure B.33 in Ref. [5]. In this model, there are two tasks referring to task A. Every execution of tasks can be recorded as events in log, so the cases for the execution of Fig.1 can be sequence of events "A, B, C, A" or "A, C, B, A". There is no easy way to distinguish the events "A" from

the previous task A and the ones from the latter task A, so it is difficult to reconstruct the original process model having duplicate tasks from event log.

Table 1 is a group of traces log generated by Fig.1. According to such traces, we know that task A is the start node of the process model, and A is the end node of the process model at the same time, and then A becomes an independent process isolated from other tasks. Fig.2 is the mined model through α algorithm for the inputted data in Table 1, and α algorithm can not support duplicate tasks construct. Duplicate tasks disturb the mined model seriously if mining algorithms can not solve this challenge.

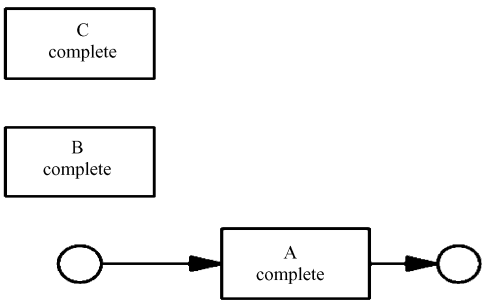

Fig.2 Mining figure 1 through α algorithm

Table 1 Traces log example of figure 1

case name	trace log of case
$trace_1$	A, C, B, A
$trace_2$	A, B, C, A
$trace_3$	A, C, B, A
$trace_4$	A, C, B, A
⋮	⋮

That is an example of the challenging problem "mining duplicate tasks", and there are some examples that came from realistic situation, such as Figure B.17 (betaSimplified) and Figure B.29 (flightCar) in Ref.[5].

According to the combination between duplicate tasks and other constructs, models having duplicate tasks can be classified. Such combinations are numerous, and the classes of duplicate tasks are fragmentary. To simplify the discussion in following sections, we roughly divided models having duplicate tasks into three major classes:

- Duplicate tasks in sequence construct;
- Duplicate tasks in parallel construct;
- Multiple duplicate tasks.

3 Distinguish duplicate tasks

We introduce a duplicate tasks treatment stage before the execution of primary process mining algorithm, and such stage will distinguish different tasks referring to a same task and consequently rename the related events. In the next step, process mining algorithms could get rid of the renamed event log that with no duplicate tasks disturbance.

Take Table 1 as an example, every trace in it has two events "A". These two "A" are an example for duplicate tasks. We notice that one of the events "A" is generated by the previous task "A" in Fig.1, whereas the other "A" is generated by the latter task "A" in it.

The target of duplicate tasks treatment is to divide all events "A" generated by different tasks into different groups, and then change different groups of events "A" into different names, such as "A_0" or "A_1".

Depending on the quality of duplicate tasks identification, we classify the results of duplicate tasks treatment into following levels:

● Perfect treatment: If the identification is correct for every event, then it means that there are no events that came from different tasks are sharing the same new name, and the number of tasks that originally referring to a same task is exactly the number of new tasks originated from the same task. To achieve this level of quality is our goal in duplicate tasks treatment stage.

● Over treatment: If there are no events that came from different tasks are appointed to a single group, but the number of groups for events that originally referring to a same task is greater than the number of different tasks that referring to the same task in expecting process model. Suppose events came from the previous "A" in Fig.1 are

divided to more than one group of events, then it is an example of over treatment. We notice that over treatment is not an error, but a kind of slightly low quality of identification.

● Wrong treatment: There are some events came from different tasks are appointed to a single group. Suppose some events came from the previous task "A" in Fig.1 and some events came from the latter "A" are appointed to the same group and refer to new task "A_0", then we will get "A_0" is the first task and the last task in expecting process mining at the same time, and this situation is contrast with there should be some tasks "B" and "C" between the start and the end node of process model. Therefore, Wrong treatment results unacceptable output data that will really mess up mined process model in the following mining stage.

● Failed treatment: If the output of duplicate tasks treatment stage is the same as original input data but there are some duplicate tasks that should be treated. Failed treatment leaves duplicate tasks to the following mining algorithms and provides no help on solving duplicate tasks.

In next section, we describe an algorithm that automatically classifies events of duplicate tasks basing on context information of events in traces. In detail, we use Nearest Neighbor clustering methods to demonstrate our idea.

4 Algorithm design

Definition 1 An event in traces log is expressed as $e_i = e_{t_i, s_i}$, in which t_i is the trace id that contains this event, and s_i is the sequence number of this event in such trace.

For example, the first "A" of $trace_1$ in Table 1 will be marked as $e_{1,1}$, and the second "A" of $trace_1$ will be marked as $e_{1,4}$.

Definition 2 The characteristic of an event is marked as $cha(e_i)$, and $cha(e_i)$ is a sequence of events, which is composed of the previous W events and the succeeded W events if there are any, in which W is the compared window parameter that control how many events get involved when we compare the characteristic of two events.

Take Table 1 as an example, when we set W as 3, $cha(e_{1,1})$ will be "C, B, A", $cha(e_{1,2})$ will be "A, B, A", $cha(e_{1,3})$ will be "A, C, A", $cha(e_{1,4})$ will be "A, C, B". Although W as 3 seems to produce characteristic as long as 6 events at most, the length of traces in Table 1 are just 4 events, and then the characteristic of events are much shorter than we expect.

Definition 3 The distance of two events is marked as $dist(e_i, e_j)$, and we call it as the distance basing on difference. $dist(e_i, e_j)$, is the number of different events in $cha(e_i)$ and $cha(e_j)$.

For example, $cha(e_{1,1})$ is "C, B, A", $cha(e_{1,4})$ is "A, C, B", and the difference is that $cha(e_{1,4})$ has an "A" at first whereas $cha(e_{1,1})$ has an "A" at last, so $dist(e_{1,1}, e_{1,4})$ is 2.

Algorithm Input traces log $T = \{trace_i\}$, divide events came from duplicage tasks into different groups, and change the name of each group to distinguish duplicate tasks.

(1) For all events referring to a same task in T, do the following steps, until all events have been considered.

(2) Mark all events referring to the same task named as "l" in T as $\{e_{(l)i}\}$. Set $i = 1$ and $k = 1$. Assign event $e_{(l)1}$ to cluster c_1.

(3) Set $i = i + 1$. Find nearest neighbor of event $e_{(l)i}$ among the events already assigned to clusters. Let d_m denote the distance basing on difference from $e_{(l)i}$ to its nearest neighbor. Suppose the nearest neighbor is in cluster c_m.

(4) If d_m is no greater than Th, then assign event $e_{(l)i}$ to cluster c_m, where Th is the threshold specified by the user. Otherwise set $k = k + 1$ and assign $e_{(l)i}$ to a new cluster c_k.

(5) If every event in $\{e_{(l)i}\}$ has been considered then go to next step, else go to step (3).

(6) If k is greater than 1, then rename the events assigned to cluster c_k with the original name plus k postfix. For example, the original name of event $e_{(A)1,1}$ is "A", and if $e_{(A)1,1}$ assigned to cluster c_1, then $e_{(A)1,1}$ will be renamed to "A_1".

In such algorithm, we set two parameters. One is W, it controls how many previous and succeeded events are used when composing the characteristic of an event. The other one is Th, it controls the distance that an event will be assigned to an existent cluster.

The most of algorithms with duplicate tasks support mainly consider just one previous and one succeeded event, such as Li's algorithm in Ref. [7, 8]. In contrast, when we set a suitable value to parameter W, the algorithm described above could cover long-range information, which information helps to avoid the limitation of just using local reference.

Additionally, we produce the measure of distance basing on difference as definition 3. Such definition converts the relationships between events into a numerical form. Considering there are many existent clustering and classification algorithms are designed towards numerical data, definition 3 can help transfer such algorithms into duplicate tasks area, which will improve the performance of it. To demonstrate the idea of duplicate tasks clustering, we use Nearest Neighbor clustering here, because such clustering algorithm needs no future knowledge of data and is easy to implement.

5 Experiment

To test our algorithm, we choose 7 typical process models having duplicate tasks from Ref. [5]. Such process models include duplicate tasks in sequence construct, duplicate tasks in parallel construct, and multiple duplicate tasks. The characteristic of each model and the result of duplicate tasks treatment are shown in Table 2.

Table 2　Results of experiments

model	characteristic	best parameters	result
B.29: flightCar	duplicates in parallel	W:1, Th:2	perfect treatment
B.33: herbstFig5p1AND	duplicates in sequence	W:3, Th:1	over treatment for A
B.35: herbstFig5p1OR	duplicates in sequence	W:2, Th:1	over treatment for A
B.37: herbstFig5p19	duplicates in sequence	W:1, Th:2	over treatment for C
B.45: herbstFig6p31	duplicates in sequence	W:2, Th:2	perfect treatment
B.55: herbstFig6p38	duplicates in parallel	W:3, Th:2	over treatment and wrong treatment for A
B.65: herbstFig6p9	two duplicates in sequence	W:2, Th:1	perfect treatment for C, over treatment for A_1, D, E

"W" in parameters column means compare window; "Th" in parameters column means threshold in algorithm.

In detail, we use CPN Tools 2.2.0[10] to illustrate the expecting process model in Colored Petri nets (CP-nets), and simulate 300 trace cases for every process model, and then use ProM Import Framework 4.0[11] to transfer the simulated data into Mining XML (MXML) format, which is a common log format for different process mining tools. The detailed tutorial for process log simulation is described in Ref. [12].

We implement the duplicate tasks treatment algorithm of this paper in Python programming language, and apply the code on generated process log through Python 2.5.1 on Windows platform. Such code reads process log in MXML format, and outputs the transferred traces into another file with the same format.

At last, we open the transferred log in ProM Framework 4.1[13], and use Alpha process mining plugin to mine the log. We choose α algorithm, because α algorithm supports the discovery of any workflow represented by a so-called SWF-net(Sound Workflow nets)[9], so the process mining stage following the duplicate tasks treatment stage could well reconstruct the expecting process model. A more important reason for choosing α algorithm is that such algorithm is noise sensitive, so any mistake in duplicate tasks treatment stage will result in serious

transformation for mined model, and such feature can be a good test for the performance of duplicate tasks treatment stage.

As described in section 2, although over treatment is not a perfect result, it is not a wrong or failed result either. Fig.1, is one of our experiment process models. When we set parameter W as 3, Th as 1, we get mined process model as Fig.3. Compare Fig.1 and Fig.3, the previous task "A" in Fig.1 is divided into "A_0", "A_2" in Fig.3, and the latter task "A" in Fig.1 is divided into "A_1", "A_3" in Fig.3. Obviously, the result in Fig.3 is not an optimal model, but it is well consistent with the original trace log. If we examine Fig.3 carefully, we will consider whether "A_0" and "A_2" can be merged into a single task, and whether "A_1" and "A_3" can be merged into a single task. Furthermore, the other algorithms trying to tackle duplicate tasks seldom assure perfect preciseness either. Therefore, we consider over treatment is acceptable result instead of wrong one.

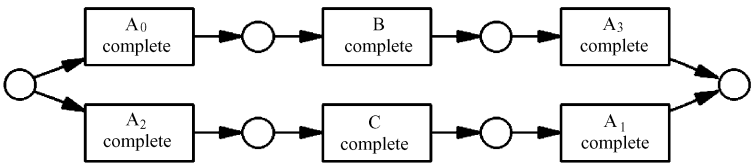

Fig.3 Discovered process model through duplicate task treatment (herbstFig5p1AND)

- For the wrong treatment in process model "B.55: herbstFig6p38", some events came from task "A" and "A_1" are merged into the same new name "A_2", and such "A_2" is isolated from the mined model. The other part of the mined model is consistent with inputted trace log.
- For model "B.33: herbstFig5p1AND" and "B.35: herbstFig5p1OR", the main structure of the two models are parallel, and the models are too short that can not provide enough information to distinguish duplicate tasks. To avoid failed treatment, the best parameters lead to over treatment for "A".
- For model "B.37: herbstFig5p19", the previous and succeeded part of task "C" are both parallel constructs. Such parallels disturb the precise identification for "C" to be a non duplicate task.
- For model "B.65: herbstFig6p9", tasks "A_1", "D", "E" are disturbed by the two adjacent parallel constructs.

To summarize the experiment result of Table 2, there are 2 perfect treatments, 4 over treatments and 1 wrong treatment. Such experiments prove that the duplicate tasks treatment mechanism and the algorithm in this paper are validated. The improvement of clustering part of the algorithm will alleviate the over treatment situation. And the computation of distance basing on difference consumes more than 85% computing time through the whole execution of algorithm.

6 Conclusion

Process mining is the process model rediscovery approach from monitored data. It is difficult because of the lack of future knowledge. For the same reason, the tackle of duplicate tasks is difficult too. This paper designed a distance measure basing on the difference of event context, and such kind of distance allows us to apply various machine learning technology on dealing with duplicate tasks. We used Nearest Neighbor clustering to demonstrate this idea. Furthermore, we isolated the treatment of duplicate tasks from whole process mining approach, so the method in this paper is well compatible with all process mining algorithms that can not support duplicate tasks. And we proved our algorithm through simulation experiments for 7 typical process models having duplicate tasks.

There are some further problems valuable to consider in future works. First, the Nearest Neighbor clustering is a simple algorithm that is chosen to prove our concept. A better clustering algorithm will improve the performance of

duplicate tasks treatment, and reduce the possibility of over treatment situation. Second, this paper set two parameters for algorithm manually, how to provide an automatic method to choose suitable parameters is another problem needed to consider. One possible direction is to examine the statistical distribution of distance between events referring to a same task.

References

[1] Aalst WMP, Weijters AJMM. Process mining: A research agenda. *Computers in Industry*, 2004, 53(3): 231 ~ 244

[2] Dumas M, Aalst WMP, Hofstede AH. Process Aware information systems: bridging people and software through process technology. Wiley-Interscience, 2005

[3] Aalst WMP, Hee KM. Workflow management: models, methods, and systems. Cambridge, MA: MIT Press, 2002

[4] Workflow Management Coalition. WFMC home page. [2005-11-15]. http://www.wfmc.org

[5] Medeiros AKA. Genetic process mining: [Ph.D. Thesis]. Eindhoven: Universiteit Eindhoven, 2006

[6] Song JL, Luo TJ, Chen S. Behavior pattern mining: apply process mining technology to common event logs of information systems. In: 2008 IEEE International Conference on Networking, Sensing and Control. Sanya, China, 2008. 1800 ~ 1805

[7] Li JF, Liu DY, Yang B. Process mining: An extended α-algorithm to discovery duplicate tasks. *Chinese Journal of Computers*, 2007, 30(8): 1436 ~ 1445(in Chinese)
李嘉菲, 刘大有, 杨　博. 过程挖掘中一种能发现重复任务的扩展 α 算法. 计算机学报, 2007, 30(8): 1436 ~ 1445

[8] Li JF, Liu DY, Yu WJ. Process mining algorithm to discover duplicate tasks. *Journal of Jilin University* (*Engineering and Technology Edition*), 2007, 37(1): 107 ~ 110(in Chinese)
李嘉菲, 刘大有, 于万钧. 一种能发现重复任务的过程挖掘算法. 吉林大学学报(工学版), 2007, 37(1): 107 ~ 110

[9] Aalst WMP, Weijters AJMM, Maruster L. Workflow mining: discovering process models from event logs. *IEEE Transactions on Knowledge Data Engineering*, 2004, 16(9): 1128 ~ 1142

[10] CPN tools homepage. [2007-10-15]. http://www.daimi.au.dk/CPNTools/

[11] ProM import framework homepage. [2007-05-16]. http://is.tm.tue.nl/ ~ cgunther/dev/promimport/

[12] Medeiros AKA, Günther CW. Process mining: using CPN tools to create test logs for mining algorithms. In: Jensen K(ed). Proceedings of the 6th Workshop on the Practical Use of Coloured Petri Nets and CPN Tools (CPN 2005). Aarhus, Denmark, 2005. 177 ~ 190

[13] ProM framework homepage. [2007-05-09]. http://prom.sourceforge.net/

一种利用聚类思想识别重复任务问题的处理方法

宋进亮　罗铁坚　陈　肃　刘　伟

(中国科学院研究生院,北京 100049)

摘　要　流程挖掘是一种从实际业务执行日志中发现结构化流程信息的过程.流程挖掘技术广泛应用于业务流程的发现和辅助建模过程中,并能够通过差异分析的方法帮助改进已有业务流程.如何处理流程模型中的重复任务,是流程挖掘技术的一个关键问题.提出了一个在标准流程挖掘算法执行之前进行的重复任务处理阶段,这一重复任务处理方法可以很好地兼容目前已有的各种流程挖掘算法,使之能处理重复任务.并提出了一种能够将事件记录上下文信息的差别数值化的距离度量定义,使用这种度量能够利用聚类方法来识别输入数据中的重复任务.最后利用典型的带有重复任务的流程模型,对所提出的处理方法进行模拟实验,并取得了良好的实验效果.

关键词　流程挖掘, 重复任务, 聚类