International Journal of Science and Engineering Applications
Volume 12-Issue 10, 34 – 38, 2023, ISSN:- 2319 - 7560
DOI: 10.7753/IJSEA1210.1011

A Two-Stage Process Discovery Algorithm Capable of Identifying Duplicate Tasks

Xuan SU
School of Computer Science and Technology
Shandong University of Technology
Zibo, 255000, China

Abstract:Process Mining is a novel technology for discovering process-related information from business data, aiming to discover, perform compliance checks, and improve business processes. The discovery of business processes is the first step in process mining. Due to the inability of traditional algorithms to identify task instances with the same name but different execution semantics, i.e., duplicate tasks, this paper proposes a new process discovery method. Its main principle is to leverage the feature of transition systems that can track the pre- and post-execution states of tasks. This feature is used to distinguish tasks with the same name under different execution states. Subsequently, it constructs a set of directed flow relations that describe the predecessor and successor relationships between tasks. Then, it employs an inductive discovery algorithm, Inductive Miner, to transform the set of directed flow relations into a process tree for identifying complex relationships between concurrent, choice, and other tasks, which are then transformed into a Petri net. Experimental results demonstrate that this method not only identifies duplicate tasks but also reduces the number of implicit transitions in the model. This significantly improves the accuracy of the discovered process model compared to the Inductive Miner.

Keywords: process mining; business process discovery; duplicate task identification; transition system

1. INTRODUCTION

Business process discovery is one of the most challenging research areas within the field of Process Mining (PM)[1-3]. Its primary goal is to construct a process model that reflects real-world business processes based on event logs containing actual business execution information. This model extends beyond the control flow dimension, describing the sequence of tasks, to also incorporate additional attributes related to task execution. These attributes can encompass resources, time, roles, and organizational aspects.

The term "duplicate tasks" refers to tasks with the same name but different conditions or objectives during their execution within a business process. In other words, these are instances of the same task with varying contextual significance. Currently, existing process discovery algorithms do not support the identification of duplicate tasks[4-6].

To address this challenge, this study introduces the concept that the context of duplicate tasks[7-9], including the sets of tasks executed "before" and "after," task multisets, and task sequences, is different. In other words, the context of duplicate tasks is distinct. To achieve this distinction, a transition system is introduced to identify the states before and after task execution. In the transition system, tasks are represented as events, and the states before and after task execution are represented using sets, multisets, or sequences. Leveraging this characteristic, the study extracts the order relationships between input transitions and output transitions under the same state. This allows for the construction of a set of directed flow relations that describe the "predecessor" and "successor" relationships among duplicate tasks.

Subsequently, the advantages of the inductive discovery algorithm (Inductive Miner, IM)[10] are utilized to construct a process tree based on this set of directed flow relations. This enables the discovery of more complex task relationships within the set of direct follow relations. Finally, the output is a Petri net that can recognize duplicate tasks.

Experimental results demonstrate that this method significantly improves precision compared to traditional IM algorithms. In other words, it enhances the model's ability to accurately describe event logs.

The remainder of this paper is organized as follows. Section 2 discusses the related work. Section 3 introduces background knowledge. Section 4 introduces two-stage business process discovery algorithm. Section 5 describes the tool implementation. Section 6 describes the data set used in the experiments, introduces the experiments and shows the results of the evaluation. Finally, Section 7 draws conclusions and points our future research scope.

2. RELATED WORK

In the domain of Process Mining (PM), there are primarily four categories of business process discovery algorithms:

1. Direct Arithmetic Methods:

The first category involves direct arithmetic methods that analyze the sequential relationships between tasks. Examples of this category include the Alpha series algorithms, as seen in references [11-12].

2. Two-Stage Methods:

The second category encompasses two-stage methods[13]. These methods construct a low-level model, such as a transition system or hidden Markov model, and then transform it into a high-level model capable of describing complex relationships between tasks. An example is the Multi-phase mining algorithm.

3. Intelligent Computing Methods:

The third category employs intelligent computing methods, utilizing machine learning, deep learning[17], reinforcement learning, genetic algorithms, and continuous iterations to fit a process model that aligns with the event log. An instance is the Genetic Miner algorithm.

International Journal of Science and Engineering Applications
Volume 12-Issue 10, 34 – 38, 2023, ISSN:- 2319 - 7560
DOI: 10.7753/IJSEA1210.1011

4. Local Methods:

The fourth category, known as local methods, focuses on discovering rules and frequent patterns between tasks and task sets rather than covering a complete process from start to finish. An example is the Declare model discovery method based on temporal logic language.

A transition system is a specific type of Finite State Machine (FSM). Its minimal components consist of the state before transition triggering, the transition itself, and the state after transition triggering. Each transition system defaults to the premise that any event can trigger a transition. After an event is triggered, a new state must be generated, and any state originates from the same initial state. Various characteristics of transition systems include subsets of different types of states being subsumed into different domains (regions), providing a method for synthesizing Petri nets.

Inductive Miner (IM) algorithm emerged in 2013 and is currently the only algorithm capable of making the discovered model perfectly match the event log. Therefore, it is often used as part of the business process discovery architecture. For instance, it discovers hierarchical business processes and can also discover cross-organizational business processes. The main principle of IM involves analyzing the sequential relationships between tasks and task sets to construct a process tree. This tree is then transformed into a Petri net. The process tree is a block-structured workflow net, where the leaf nodes represent tasks, and parent nodes are operators describing relationships between sub-trees. While IM ensures fitness of the model, it forces the model to record behaviors not present in the event log, sacrificing precision. Additionally, it cannot distinguish duplicate tasks, which is a significant factor leading to irrelevant behavior in the model.

To address these drawbacks, this paper introduces the use of transition systems to track task execution states. By distinguishing duplicate tasks based on the differing execution statuses of tasks with the same name, and leveraging the IM algorithm, the paper aims to enhance precision while ensuring fitness.

3. BACKGROUND KNOWLEDGE

Definition 1: Event. An event is the smallest element composing log data, representing an instance of a task's execution in a business system. It is denoted as e = (a, cid, resource, start, end, other), where A is a set of activity names, and 'a' represents the activity executed in the event. 'cid' denotes the unique identifier of the running instance, 'resource' indicates the resources required for the event's execution, and 'start' and 'end' represent the start and end times of the event's execution. Additionally, events can have more detailed attributes in different scenarios, such as education or healthcare, where event attributes may vary. 'Other' refers to additional properties. Let N be the set of events containing attribute sets.

Definition 2: Classifier. If there exists n ∈ N, for any event e in the set, #n(e) represents the value of its attribute n. Let UC be the set of cases, UA be the set of tasks, UL be the set of lifecycles, and UT be the set of timestamps. It is assumed that for any event e, it contains the following attributes: #case(e) ∈ UC, representing the case to which event e belongs (each event belongs to one case only); #act(e) ∈ UA, representing the activity name of event e; #trans(e) ∈ UL, containing lifecycle-related information for event e; #time(e) ∈ UT, indicating the timestamp when event e occurred.

Definition 3: Trace, Case, Event Log. A trace is a finite sequence of events. Let * be the set of all finite event sequences defined over a set, and a case is a finite sequence of events *, denoted as =<e1, e2, e3, ..., e||>, satisfying the conditions that each event can occur only once, i.e., for $1 \le j < k \le ||$, (j)(k), and every event in a case has the same case identifier, i.e., e1, e2, #case(e1) = #case(e2).

An event log is a collection of finite event sequences, defined as L*. Here, L = {1, 2, ..., |L|}.

Definition 3: Multiset. A multiset is a collection in which elements can appear multiple times. Let m = [p3, q2] be a multiset defined over the set S = {p, q}, where m ⊆ B(S), m(p) = 3, and m(q) = 2 represent the number of occurrences of elements. B(S) denotes the universal set of multisets over S. Multisets are used not only to track the execution status of tasks but also for modeling event logs. A trace appearing in different cases can be represented using multisets.

Definition 4: Petri Net. A Petri net is used to represent the evolution of processes and can capture the control flow relationships between entities. It consists of a triple N = (P, T, F), where T represents a finite set of transitions, P represents a set of places, and F ⊆ (P × T) ∪ (T × P) is the set of arcs representing control flow in the Petri net. A marking (M) of a Petri net (N, M) is a multiset of tokens, where tokens are represented by black dots. M ⊆ B(P) indicates a multiset defined on places, representing the tokens held in the places. τ is an implicit transition introduced to ensure modeling correctness and can represent the silent execution of tasks.

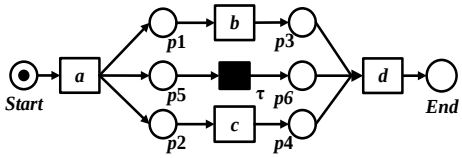

Figure. 1 The Petri net

Definition 5: Firing Rule. The firing rule describes the conditions under which transitions in a Petri net can occur. For x ∈ P ∪ T, •x = {y | y ∈ P ∪ T ∧ (y, x) ∈ F} is the pre-set of x, and x• = {y | y ∈ P ∪ T ∧ (x, y) ∈ F} is the post-set of x. A transition t in a Petri net N with a marking m satisfies the following condition: for any place p ∈ •t: m(p) ≥ 1. In such cases, transition t is enabled, can fire, and produces a new marking, denoted as (N, m)[t→(N, m'). If all input places of a transition contain sufficient tokens, the transition is enabled, can fire, and consumes one token from each input place, producing new tokens in the output places. In Figure 1, assuming the initial marking is a multiset [start4], after firing 'a,' it leads to [start3, p1, p2], 'a' is still enabled, and after firing 'a' again, it results in [start2, p12, p22], and so on. 'b' and 'c' become enabled simultaneously.

4. TWO-STAGE BUSINESS PROCESS DISCOVERY ALGORITHM

The algorithm proposed in this article essentially transforms the transition system into a collection of direct follow relations. It then applies the inductive miner algorithm to discover a model. The steps of Inductive Miner is shown in Figure 2.

International Journal of Science and Engineering Applications
Volume 12-Issue 10, 34 – 38, 2023, ISSN:- 2319 - 7560
DOI: 10.7753/IJSEA1210.1011

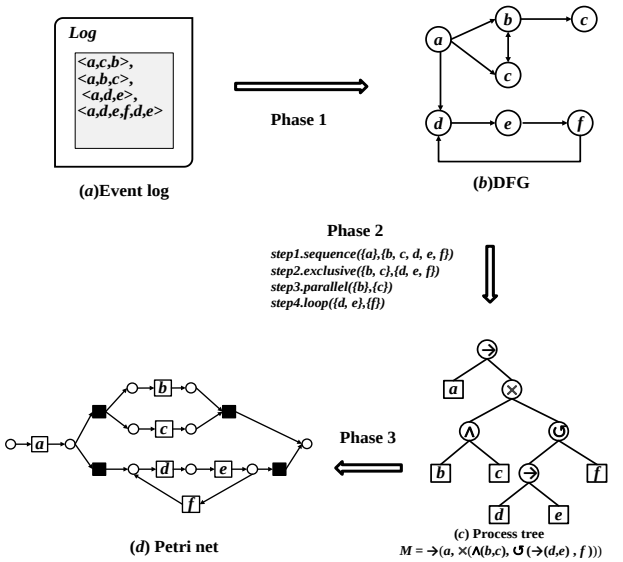

Figure. 2 The steps of Inductive Miner

Following the rules of the inductive miner algorithm, the process tree is further transformed into a Petri net. Additionally, let L={[1=<A,B,C,D>, 2=<A,B,C,A,D>]}.The primary principle of this method in the article involves traversing each trace in the event log, simulating the execution of the trace, and tracking the past and future states of each task within the trace. It assumes that the starting point of each trace's execution in the transition system is an empty set, denoted as `start`. Each task is represented as an event `e`. If, after executing the current task, the "future" state of the current event already exists in the transition system, the algorithm proceeds to the next task. If it does not exist, a new state is added to the transition system.The transition system generated from L is shown in Figure 3

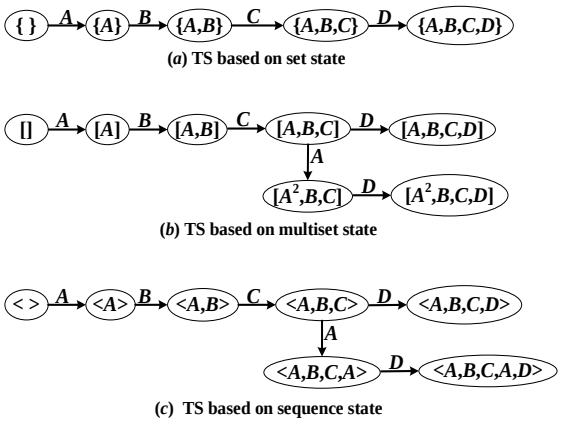

Figure. 3 The transition system generated from event log L

Because the collection of direct follow relations serves as the input for the inductive mining algorithm to discover the process tree, the transformation of the transition system into a collection of direct follow relations is the most crucial part of the two-stage discovery algorithm in this article. The main implementation principle involves two steps, firstly, traversing the set of transitions to assign different numbers to events or tasks with the same names in different transitions. This is done to distinguish events with the same name but different pre and post-execution states. Secondly, Traversing the set of states in the transition system, where the input events for a state become the predecessors of the direct follow relations, and the output events become the successors.Finally, the execution example of the two-stage discovery algorithm is shown in Figure 4.

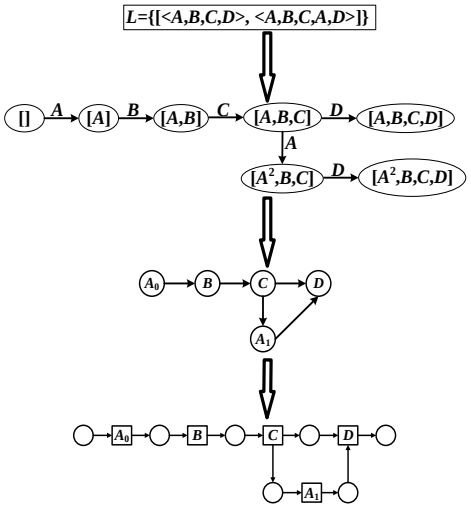

Figure. 4 Execution example of the two-stage algorithm

5. TOOL IMPLEMENTATION

In this experiment, we use a laptop with a 2.70 *GHz CPU*, *Windows* 10 Professional, *Java SE* 1.8.0_281 (64-bit) , Python 3.7.6 (64-bit) and allocate 12 *GB* of *RAM*. In addition, the drawing software *Origin* 2021 *Pro* version is used to show the experimental results.

The open source process mining tool platform *ProM* provides a fully pluggable experimental environment for process mining. It can be extended by adding plugins and currently contains more than 1600 plugins. The tool and all plugins are open source. Set coverage sampling approach proposed in this paper has been implemented in *ProM* platform as plugin, which called *Two-stage process discovery algorithm capable of identifying duplicate tasks*. The snapshot of this tool is shown in Figures 5. It takes an original event log as input and outputs a model.

Figure. 5 The instance of *ProM* plugin

6. EXPERIMENTAL EVALUATION

6.1 Experimental data sets

The data used in this article is sourced from reference [16]. In this chapter, online learning of "Microprocessor System Design" is used as a case study. To ensure user anonymity and privacy protection, the data has been appropriately formatted and cleaned. The original logs contained data records from the client system every second.This article selected relevant attributes and presented the data in a format suitable for process mining. The data attributes include chapters, student identifiers, and identifiers for each exercise practice. Each CSV file corresponds to a specific chapter or a specific student, and each file contains several exercise practices for that session. Each exercise practice

includes specific learning behaviors, and the performance data comprises two CSV files: final grades obtained from the end-of-course exams and stage grades obtained from student homework assessments in each class.An overview of the cleaned event log is shown in Table 1. The "id" column represents a unique numerical identifier for a student participating in the online course.

Table 1. Online learning event log overview

Event log	Trace number	Event number	Event types number
Session1	77	71836	28
Session2	82	83014	38
Session3	87	60412	48

6.2 Experiment results

The fitness[14] results of both the algorithm in this chapter and the IM algorithm are 1, so no further records are made. The precision[15] comparison results are shown in Table 2. It can be observed that the algorithm proposed in this paper significantly improves the precision of the model while ensuring model quality. This ensures the use of a reliable process model for future process analysis. Additionally, it enhances the efficiency of model analysis, providing decision-makers with more valuable and accurate information.

Table 2. The algorithm evaluate results

Event log	IM	Two-stage	TS state type
session1	0.32	0.39	set
		0.42	multi-set
		0.42	sequence
session2	0.25	0.54	set
		0.52	multi-set
		0.49	sequence
session3	0.47	0.61	set
		0.58	multi-set
		0.62	sequence

7. CONCLUSIONS

This article introduces a novel process discovery method, the main principle of which is to leverage the capability of tracking the pre and post-execution states of tasks using transition systems. This feature helps distinguish tasks with the same name under different execution states. Consequently, it constructs a collection of directed flow relationships to describe the predecessor and successor relationships between tasks. Subsequently, it employs the inductive mining algorithm (Inductive Miner, IM) to transform the collection of directed flow relationships into a process tree. This process tree is used to identify complex relationships between tasks, such as concurrency and choice, and then further transform them into a Petri net.Experimental results demonstrate that this method not only identifies duplicate tasks but also reduces the inclusion of implicit transitions in the model, resulting in significantly improved accuracy compared to the Inductive Miner algorithm. In the future, further optimization of the identification of duplicate tasks within the transition system from a contextual perspective could lead to even more accurate process models.

8. ACKNOWLEDGMENTS

This paper is supported by the Taishan Scholars Program of Shandong Province (No.ts20190936, tsqn201909109), the Natural Science Excellent Youth Foundation of Shandong Province (ZR2021YQ45), and the Youth Innovation Science and Technology Team Foundation of Shandong Higher School (No.2021KJ031).

REFERENCES

[1] VAN DER AALST W.M.P. Process Mining: Discovery, Conformance and Enhancement of Business Processes[M]. Springer Publishing Company, Incorporated, 2011.

[2] DOGAN G .Process mining: data science in action[J].Computing reviews, 2017, 58(6):337-337.

[3] LIU Cong, DUAN Hua, ZENG Qingtian, et al. Towards comprehensive support for privacy preservation cross-organization business process mining[J]. IEEE Transactions on Services Computing, 2019,12(4):639-653.

[??] PEDRO J D S , CORTADELLA J. Discovering Duplicate Tasks in Transition Systems for the Simplification of Process Models[J].Transition Systems for the Simplification of Process Models. 2017:108-124.

[??] LU Xixi, FAHLAND D., VAN DER BIGGELAAR F., et al. Handling duplicated tasks in process discovery by refining event labels. International Conference on Business Process Management, vol, 9850. Springer, 2016:90-107.

[6] VáZQUEZ BARREIROS B., MUCIENTES M., LAMA M.. Enhancing discovered processes with duplicate tasks. Information Sciences, vol. 373, 2016:369-387.

[7] VANDEN BROUCKE S. K., DE WEERDT J.. Fodina: a robust and flexible heuristic process discovery technique. Decision Support systems, vol. 100, 2017:109-118,.

[??] DUAN Chenchen, WEI Qingjie. Process Mining of Duplicate Tasks: A Systematic Literature Review. 2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA), 2020, 778-784.

[??] ZHANG Yuheng. Research on Conformance Checking Method for Duplicate Tasks in Process[D],Master Thesis, Chongqing University of Posts and Telecommunications,2022,

[10] LEEMANS S J J, FAHLAND D, VAN DER AALST W.M.P.. Discovering block-structured process models from event logs-a constructive approach[C]. International conference on applications and theory of Petri nets and concurrency. 2013: 311-329.

[VAN DER AALST W.M.P., WEIJTERS T, MARUSTER L. Workflow mining: Discovering process models from event logs[J]. IEEE transactions on knowledge and data engineering, 2004, 16(9): 1128-1142.

[WEN Lijie, WANG Jianhua, VAN DER AALST W.M.P., et al. A novel approach for process mining based on event types[J]. Journal of Intelligent Information Systems, 2009,32(2): 163-190.

[13] VAN der Aalst, W.M.P. ; RUBIN, V. ; Verbeek, H.M.W. et al. / Process Mining: A Two-Step Approach to Balance Between Underfitting and Overfitting. BPMcenter.org, 2008. 40 p. (BPM Center Report; No. BPM-08-01).

[ADRIANSYAH A., B. F. VAN DONGEN,VAN DER AALST W.M.P.. Conformance Checking using Cost-Based Fitness Analysis.IEEE International Enterprise Distributed Object Computing Conference.2011:55-64.

[WEERDT J, BACKER , VANTHIENEN J, BAESENS B. A robust F-measure for evaluating discovered process models. Washington, D. C., USA: IEEE ,2011: 148–155.

[VAHDAT M. A learning analytics approach to correlate the academic achievements of students with interaction data from an educational simulator[M]//LUCA O, DAVIDE A.Design for teaching and learning in a networked world: Discovery Science. Berlin: Springer International Publishing, 2015:352-366.

[BANNERT M, REIMANN P, SONNENBERG C. Process mining techniques for analysing patterns and strategies in students' self-regulated learning[J]. Metacognition & Learning, 2014(8):161-185.