Hung Son Nguyen
Quang-Thuy Ha · Tianrui Li
Małgorzata Przybyła-Kasperek (Eds.)

LNAI 11103

# Rough Sets

## International Joint Conference, IJCRS 2018
## Quy Nhon, Vietnam, August 20–24, 2018
## Proceedings


Springer

**Lecture Notes in Artificial Intelligence 11103**

Subseries of Lecture Notes in Computer Science

More information about this series at http://www.springer.com/series/1244

Hung Son Nguyen · Quang-Thuy Ha
Tianrui Li · Małgorzata Przybyła-Kasperek (Eds.)

# Rough Sets

## International Joint Conference, IJCRS 2018
## Quy Nhon, Vietnam, August 20–24, 2018
## Proceedings

Springer

*Editors*
Hung Son Nguyen
University of Warsaw
Warsaw
Poland

Quang-Thuy Ha
Faculty of Information Technology
Vietnam National University
Hanoi
Vietnam

Tianrui Li
School of Information Science
Southwest Jiaotong University
Chengdu
China

Małgorzata Przybyła-Kasperek
Institute of Computer Science
University of Silesia
Sosnowiec
Poland

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Artificial Intelligence
ISBN 978-3-319-99367-6 ISBN 978-3-319-99368-3 (eBook)
https://doi.org/10.1007/978-3-319-99368-3

Library of Congress Control Number: 2018951242

LNCS Sublibrary: SL7 – Artificial Intelligence



This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

# Preface

The proceedings of the 2018 International Joint Conference on Rough Sets (IJCRS 2018) contain the results of the meeting of the International Rough Set Society held at the International Centre for Interdisciplinary Science and Education (ICISE) and the University of Quy Nhon in Quy Nhon, Vietnam, during August 2018.

Conferences in the IJCRS series are held annually and comprise four main tracks relating the topic rough sets to other topical paradigms: rough sets and data analysis covered by the RSCTC conference series from 1998, rough sets and granular computing covered by the RSFDGrC conference series since 1999, rough sets and knowledge technology covered by the RSKT conference series since 2006, and rough sets and intelligent systems covered by the RSEISP conference series since 2007. Owing to the gradual emergence of hybrid paradigms involving rough sets, it was deemed necessary to organize Joint Rough Set Symposiums, first in Toronto, Canada, in 2007, followed by symposiums in Chengdu, China in 2012, Halifax, Canada, 2013, Granada and Madrid, Spain, 2014, Tianjin, China, 2015, where the acronym IJCRS was proposed, continuing with the IJCRS 2016 conference in Santiago de Chile and IJCRS 2017 in Olsztyn, Poland.

The IJCRS conferences aim at bringing together experts from universities and research centers as well as from industry representing fields of research in which theoretical and applicational aspects of rough set theory already find or may potentially find usage. They also become a place for researchers who want to present their ideas to the rough set community, or for those who would like to learn about rough sets and find out if they can be useful for their problems.

This year's conference, IJCRS 2018, celebrated the 20th anniversary of the first international conference on rough sets called RSCTC, which was organized by Lech Polkowski and Andrzej Skowron during June 22–26, 1998, in Warsaw, Poland. On this occasion, we listened to a retrospective talk delivered by Andrzej Skowron, who summarized the successes of this field and showed directions for further research and development.

IJCRS 2018 attracted 61 submissions (not including invited contributions), which underwent a rigorous reviewing process. Each accepted full-length paper was evaluated by three to five experts on average. The present volume contains 45 full-length regular and workshop submissions, which were accepted by the Program Committee, as well as six invited articles.

The conference program included five keynotes and plenary talks, a fellow talk, eight parallel sessions, a tutorial, the 6th International Workshop on Three-way Decisions, Uncertainty, and Granular Computing, and a panel discussion on rough sets and data science.

The chairs of the Organizing Committee also prepared the best paper award and the best student paper award. From all research papers submitted, the Program Committee

nominated five papers as finalists for the award and, based on the final presentations during the conference, selected the winners.

We would like to express our gratitude to all the authors for submitting papers to IJCRS 2018, as well as to the members of the Program Committee for organizing this year's attractive program.

We also gratefully thank our sponsors: Vietnam National University in Ho Chi Minh City, for providing the technical support and human resources for the conference; the University of Quy Nhon, for sponsoring the reception and the conference facilities during the first day and the last day; Ton Duc Thang University, for sponsoring the pre-conference workshops on rough sets and data mining.

The conference would not have been successful without support received from distinguished individuals and organizations. We express our gratitude to the IJCRS 2018 honorary chairs, Andrzej Skowron, Huynh Thanh Dat, and Do Ngoc My, for their great leadership. We appreciate the help of Dinh Thuc Nguyen, Nguyen Tien Trung, Quang Vinh Lam, Quang Thai Thuan, Thanh Tran Thien, Luong Thi Hong Cam, Giang Thuy Minh, Phung Thai Thien Trang, Dao Thi Hong Le, Hung Nguyen-Manh, and all other representatives of Vietnam National University in Ho Chi Minh City and Quy Nhon University, who were involved in the conference organization. We would also like to thank Marcin Szeląg, Sinh Hoa Nguyen, and Dang Phuoc Huy, who supported the conference as tutorial, workshop, and special session chairs. We acknowledge the significant help from Khuong Nguyen-An, Tran Thanh Hai, Ly Tran Thai Hoc, and Marcin Szczuka provided at various stages of the conference publicity, website, and material preparation.

We are grateful to Tu Bao Ho, Hamido Fujita, Hong Yu, Andrzej Skowron, Piero Pagliani, and Mohua Banerjee for delivering excellent keynote and plenary talks and fellow talks. We thank Dominik Ślęzak and Arkadiusz Wojna for the tutorial. We are thankful to Hong Ye, Mohua Banerjee, Mihir Chakraborty, Bay Vo, and Le Thi Thuy Loan for the organization of workshops and special sessions.

Special thanks go to Alfred Hofmann of Springer, for accepting to publish the proceedings of IJCRS 2018 in the LNCS/LNAI series, and to Anna Kramer for her help with the proceedings. We are grateful to Springer for the grant of 1,000 Euro for the best paper award winners. We would also like to acknowledge the use of EasyChair, a great conference management system.

We hope that the reader will find all the papers in the proceedings interesting and stimulating.

August 2018

Hung Son Nguyen
Quang-Thuy Ha
Tianrui Li
Małgorzata Przybyła-Kasperek

# Contents

# A New Trace Clustering Algorithm Based on Context in Process Mining

Hong-Nhung Bui[1,2(✉)], Tri-Thanh Nguyen[1], Thi-Cham Nguyen[1,3], and Quang-Thuy Ha[1]

[1] Vietnam National University, Hanoi (VNU), VNU-University of Engineering and Technology (UET), 144, Xuan Thuy, Cau Giay, Hanoi, Vietnam
nhungbth@hvnh.edu.vn, {ntthanh,thuyhq}@vnu.edu.vn, nthicham@hpmu.edu.vn
[2] Banking Academy of Vietnam, 12, Chua Boc, Dong Da, Hanoi, Vietnam
[3] Hai Phong University of Medicine and Pharmacy, 72A, Nguyen Binh Khiem, Ngo Quyen, Haiphong, Vietnam

**Abstract.** In process mining, trace clustering is an important technique that attracts the attention of researchers to solve the large and complex volume of event logs. Traditional trace clustering often uses available data mining algorithms which do not exploit the characteristic of processes. In this study, we propose a new trace clustering algorithm, especially for the process mining, based on the using trace context. The proposed clustering algorithm can automatic detects the number of clusters, and it does not need a convergence iteration like traditional ones like K-means. The algorithm takes two loops over the input to generate the clusters, thus the complexity is greatly reduced. Experimental results show that our method also has good results when compared to traditional methods.

**Keywords:** Event log · Process mining · Trace context · Clustering algorithm

## 1 Introduction

Most today's modern information systems have collection of data that describes all the events of the user occur during the execution of the software system so-called event logs. Event logs play an important role in modern software systems, they record information about the system in real-time including a set of events that contain several information, e.g., *case id*, *event id*, *timestamp*, *activity*, etc., Table 1 introduces some examples about an event log. The events in the same *case* are ordered by *timestamp* and have the same "*case id*". These are valuable data for managers to analyze and evaluate the company's business processes.

Process mining includes three tasks process discovery, conformance checking and enhancement is the field that allows the use of the event log data for analysis and improvement of the processes.


H. S. Nguyen et al. (Eds.): IJCRS 2018, LNAI 11103, pp. 644–657, 2018.
https://doi.org/10.1007/978-3-319-99368-3_50

The target of process discovery is to generate a process model that captures all of the behaviors found in the event log [23]. The generated model can be used to analyze what is actually applied in daily activities of the company. It can be used to verify whether the formal process is strictly followed, or to enhance the formal process.

The event log quality is an important factor in process model generation. If the event log is homogeneous and small enough, the process model is easy to analyze as one example in Fig. 1a. However, real-life event logs are extremely huge with diverse characteristics, thus, the discovered process model may be diffuse and very hard to understand as an example in Fig. 1b. To overcome this problem, clustering a complex event log into sub-logs/clusters is one of the most widely used solution. The generated model from an event sub-log will have much lower complexity [5, 7, 9–11, 15–18, 21].

**Table 1.** A fragment of the event log [23]

| Case id | Event id | Properties | | | | |
|---|---|---|---|---|---|---|
| | | Timestamp | Activity | Resource | Cost | … |
| 1 | 4423 | 30-12-2010:11.02 | Register request | Pete | 50 | |
| | 4424 | 31-12-2010:10.06 | Examine thoroughly | Sue | 400 | |
| | 4425 | 06-01-2011:15.12 | Check ticket | Mike | 100 | |
| | 4426 | 07-01-2011:11.18 | Decide | Sara | 200 | |
| | 4427 | 07-01-2011:14.24 | Reject request | Pete | 200 | |
| 2 | 4483 | 30-12-2010:11.32 | Register request | Mike | 50 | |
| | 4485 | 30-12-2010:12.12 | Check ticket | Mike | 100 | |
| | 4487 | 30-12-2010:14.16 | Examine casually | Pete | 400 | |
| | 4488 | 06-01-2011:11.22 | Decide | Sara | 200 | |
| | 4489 | 08-01-2011:12.06 | Pay compensation | Ellen | 200 | |
| 3 | 4521 | 30-12-2010:14.32 | Register request | Pete | 50 | |
| | 4522 | 30-12-2010:15.06 | Examine casually | Mike | 400 | |
| | 4524 | 30-12-2010:16.34 | Check ticket | Ellen | 100 | |
| | 4525 | 06-01-2011:09.18 | Decide | Sara | 200 | |
| | 4526 | 08-01-2011:12.18 | Reinitie request | Sara | 200 | |
| | … | | | | | |

Traditional approaches use the data mining clustering algorithms such as Agglomerative Hierarchical Clustering, K-Means, K-Modes, etc., to cluster event logs. These algorithms are designed and used in the field of data mining, they do not exploit the specific characteristics of business processes.

In this paper, we propose a new trace clustering algorithm based on a specific characteristic of process, i.e., the context of traces in a process. The contribution of the paper includes: (1) defining a new trace context; (2) introducing a context tree; (3) giving a new event log clustering algorithm. The proposed algorithm can automatically detect the suitable number of clusters, and it does not need a convergence iteration which takes lot of time. The experimental results show that our method has significant contributions to improving the efficiency and the performance time of the process discovery task.

The rest of this article is organized as follows: First, we give an overview of the process discovery. Section 3 introduces the trace context in process mining and the new trace clustering. The experimental evaluation is described in Sect. 4. Section 5 introduces the related work. Conclusions and future work are shown in the last section.

## 2 The Brief Summary of Process Discovery Task in Process Mining

**Event Logs**

An event log is the starting point of process mining. Table 1 shows a fragment of the event log related to the handling of compensation requests of an airline. There are three cases corresponds to three compensation requests. The case 1 has five events with *id* from 4423 to 4427 that are ordered by execution time, i.e., property timestamp. For example, event 4423 executes activity "register request" at "30-12-2010:11.02" occurs before event 4424 which executes activity "examine thoroughly" at "31-12-2010:10.06". Each event in event log also is described by *resources* property, i.e., the persons executing the activities or the cost of the activity.

In process mining, the "*case id*" and "*activity*" are minimum properties that can be used to represent a case. For example, *case* 1 is represented by a sequence of five activities Register request, Examine thoroughly, Check ticket, Decide, Reject request. Such a sequence of activities is called a *trace*. For the sake of simplicity for computation, each activity name is assigned by a distinct letter label, e.g., *a* denotes activity register request. Hence, the event log in Table 1 has a more compact representation shown in Table 2, e.g., *case*1 is represented by a trace $\langle a, b, d, e, h \rangle$. This representation is used for computation, such as clustering. For example, in K-means a trace is converted into a vector as the input to the algorithm.

**Table 2.** The trace in an event log (where *a* = "register request", *b* = "examine thoroughly", *c*= "examine casually", *d* = "check ticket", *e* = "decide", *f* = "reinitiate request", *g* = "pay compensation", *h* = "reject request")

| Case id | Trace |
|---|---|
| 1 | $\langle a, b, d, e, h \rangle$ |
| 2 | $\langle a, d, c, e, g \rangle$ |
| 3 | $\langle a, c, d, e, f, b, d, e, g \rangle$ |
| 4 | $\langle a, d, b, e, h \rangle$ |
| 5 | $\langle a, c, d, e, f, d, c, e, h \rangle$ |
| 6 | $\langle a, c, d, e, g \rangle$ |
| … | … |

**Process Discovery Task**

Process discovery is the first task of process mining. It takes an event log as an input data and produces a model represented in a process modeling language, e.g., Petri net (Fig. 1), which describes the behaviors recorded in the event log by applying a process discovery algorithm, e.g., α-algorithm [23].

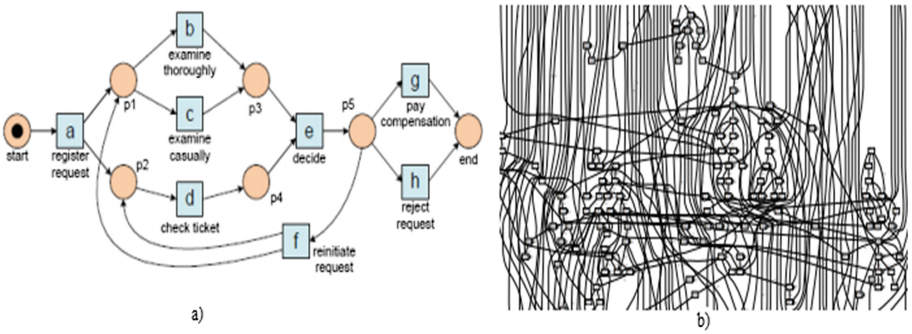


**Fig. 1.** The process model discovered from the event log by the α-algorithm

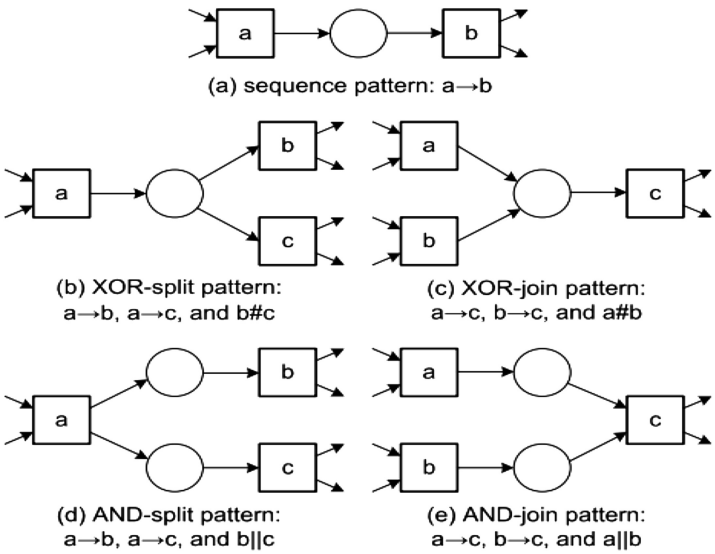


**Fig. 2.** Typical process patterns in Petri net [23]

**α-Algorithm**

The α-algorithm was one of the first process discovery algorithms. It generates the process model by reconstructing causality from a set of sequences of events in the event logs.

Given an event log of a business process $L$, α-algorithm scans $L$ to find the relationships between activities based on the execution order. There are four ordering relations, e.g., *direct succession*, *causality*, *parallel*, *choice*. Let $a, b$ are two activities in $L$.

1. Direct succession $a > b$: if some case $a$ is followed by $b$.
2. Causality $a \rightarrow b$: if activity $a$ is followed by $b$ but $b$ is never followed by $a$.
3. Parallel $a||b$: if activity $a$ is followed by $b$ and $b$ is followed by $a$.
4. Choice $a\#b$: if activity $a$ is never followed by $b$ and $b$ is never followed by $a$.

To reflect those dependencies, the Petri net has corresponding notations to connect activities as illustrated in Fig. 2.

As mentioned above, to mine easy-to-understand process models from the complex event log, the trace clustering is the effective approach. The key idea of trace clustering algorithms is to create clusters that the traces within a cluster are more similar to each other than the traces in the different clusters. Next section we introduce our proposed trace clustering algorithm.

## 3 A Context Approach to Trace Clustering

### 3.1 Context in Process Mining

In the middle of the 1990s, the context was mentioned by many researchers [2, 3, 14]. It had the important contribution to improving the performance of practical systems. Each different research fields usually have different ideas and definitions of context. It is defined as the object's location, environment, identity and execution time or object's emotional state as well as hobbies and habits of objects, etc. [12].

In process mining, the context was defined as the environment surrounding a business process, e.g., the weather conditions or holiday seasons [13]. In another study, the context was defined as the time, location, and frequency of events as well as related communication, tools, devices, or operators [22]. In [19], the context of activity $a$ was the set of two surrounding activities $xy$, i.e., $xay$, by using 3-g in an event log.

### 3.2 Trace Context

In this paper we introduce a new context definition based on the fact that each business process has a number of different procedures. For example, the credit process has procedures for personal loan, corporate loan, home loan, consumer loan, etc. Each procedure may start with a set of *common activities* which are the clue to separate traces into different clusters. In this paper we define common activities as the trace contexts.

**Definition 1.** Let $L = \{t_0, t_{1,} \ldots\}$ be an event log, where $t_i$ is a trace. Let $p$ be the longest common prefix $p$ of a trace subset, i.e., $SP = \{t \in L | t = p|d\}$, such that $|SP| > 1$, where $d$ is a sequence of activities, notation '|' in $p|d$ denotes sequence concatenation operation, then $p$ is called as a *trace context*.

### 3.3 Context Tree

Since the common prefix of traces can be represented by a prefix tree, to efficiently identify the context, we introduce a *Context-tree* based on the idea of *frequent pattern tree* (FP-tree) [8].

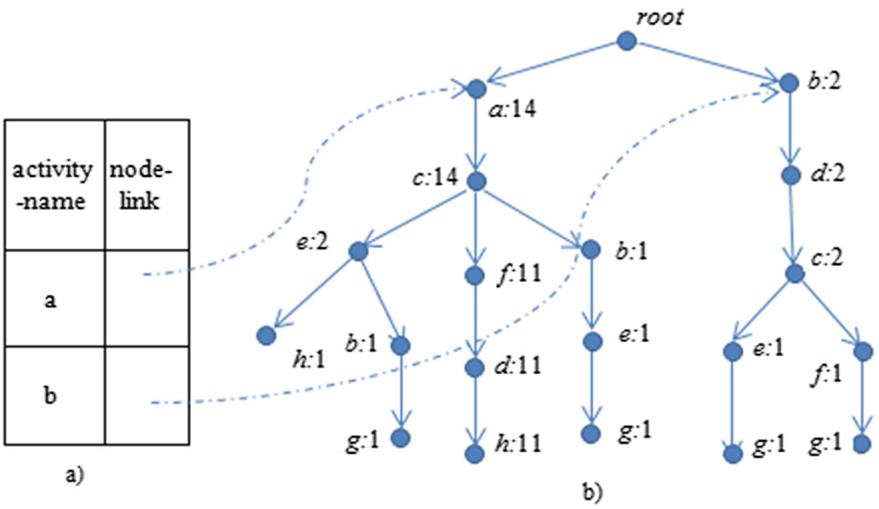


**Fig. 3.** (a) Header table; (b) The context-tree

**Definition 2.** A context tree is a tree that has:

1. One root labeled as "*root*" to form a complete tree.
2. A header table helps to access the tree faster during tree construction and traversal. Each entry in the Context-tree header table consists of two fields, (1) *activity-name*, and (2) head of *node-link* which points to the first node below the root carrying this activity.
3. Each node in the context tree consists of three fields except for the root node:

   *activity-name*: registers which activity is represented by the node;
   *count*: the number of traces that travel to this node;
   *node-link*: the pointers to its children, or null if there is none.

4. A trace in the event log is placed on a certain branch of the tree with the top- down fashion. Traces with the same prefix share a chunk of branch from the root node.

The idea is to map traces with the same prefix into the same chunk of tree branch as depicted in Fig. 3. The context tree construction procedure is described as follows:

**Algorithm 1. ContextTreeConstruction.**

```
Input:  An event log L

Output: A corresponding context tree T
1. Create a node of a Context-tree T and label it as "root",
    i.e., the root node and T = root.
2. Foreach trace t in L do
      Let t=ac|q, where ac is the first activity, and q is the
      rest of the activity sequence
      call insert_activity(ac|q, T);
   EndFor
3. Create HeaderTable and update the node-link based on the di-
rect children of the root node.
```

And the *insert_activity*(.) is defined as:

**Algorithm 2. insert_activity($ac|q$, $T$)**

```
Input: A trace in term of ac|q where ac is the first ac-
tivity, and q is the rest activities
  T is a tree node
Output: T is updated with new activities
1.  If T has a child N such that N.activity-name=ac then
      Increase N's count by 1
    Else
      Create a new node N, with its count = 1,
      Create a new node-link linked from T to N.
    EndIf
2.  If q is nonempty then
      call insert_activity(q, N) recursively.
    EndIf
```

Let L = [<*aceh*>, <*acfdh*>$^{10}$, <*acebg*>, <*acbeg*>, <*bdceg*>, <*bdcfg*>] be an event log, which includes 15 traces, the trace $\langle acfdh \rangle$ appears 10 times. The corresponding context-tree is illustrated in Fig. 3.

Mapping the context tree with the Definition 1 it is clear that, for each trace on the tree, the longest common prefix is the sequence of activities that have $count > 1$. From the context-tree in Fig. 3, the set of trace contexts of $L$ is $\{ace, acfdh, ac, bdc\}$.

If a trace is distinct from the others, then it has no context. The following procedure is responsible for identifying the context of a given trace.

**Algorithm 3. ContextDetection($ac|q$, *T, context*)**

```
Input: A trace in term of ac|q where ac is the first ac-
tivity, and q is the rest activities
       T is a context tree node
Output: The context of the trace
1.  If T is root then
      context={};
      Get the node N pointed by node-link from the
      HeaderTable of T in the entry corresponding to ac;
    Else
      Find the child node N of T that has the label ac;
    EndIf
2.  If the node N has count > 1 then
      Context = context|ac; //Concatenate a sequence
      If q is nonempty then
        call ContextDetection(q,N,context);
      EndIf
    EndIf
```

### 3.4 Context Trace Clustering Algorithm

A new trace clustering algorithm called ContextTracClus which aims at creating clusters of traces based on contexts is proposed. The algorithm consists of two distinct phases: (1) Determining trace contexts and Building clusters; (2) Adjusting clusters.

The first phase, *Determining trace contexts and Building clusters*, includes two steps.

*Step 1* builds a compact data structure called the Context-tree that stores quantitative information about activities of each trace in a event log. *Step 2* traverses the Context-tree for each trace to find its trace context, and assigns the trace to the cluster corresponding to this context. Based on the Context-tree construction process, for any trace $t$ in event log, there exists a path $p$ in the Context-tree starting from the root. The trace context of this trace is the sequence of nodes of $p$ that have $count \geq 2$. In case a trace has no context, a new cluster is created for storing this trace for later adjustment in Phase 2.

The second phase, *Adjusting clusters*, handles the case where small clusters are generated. If a cluster size, i.e., the number of traces in the cluster, is smaller than a given minimum cluster size threshold *mcs* (e.g., each cluster size should be at least 10% of the number of traces in the event log), this cluster will be added to its closest cluster. The distance between to clusters is defined as the distance between two corresponding trace contexts. In the case that a trace has no context, it will be added to the cluster whose trace context includes the maximum number of duplicate activities with this trace. The pseudo-code of the proposed algorithm, denoted ContextTracClus, is shown in Algorithm 4.

**Algorithm 4. ContextTracClus.**

```
Input:  An event log L
        A minimum cluster size threshold mcs
Output: The complete set of clusters C
Phase 1: Determining trace contexts and Building clusters
1. C = {};
2. T = ContextTreeConstruction(L);//T is the context tree
3. Foreach trace t in event log L do
      ContextDetection(t,T,context);
      If context is empty then
        Create a new cluster c;//c has no label
        Add t to c; //This cluster has only one trace
        C = C ∪ c;
      Else
        If C has no cluster labeled context then
          Create a new cluster c labeled context;
          Add t to c;
          C = C ∪ c;
        Else
          Add t to the cluster labeled context;
        EndIf
      EndIf
    EndFor
Phase 2: Adjusting clusters
4.  Foreach cluster c in C do
      If size(c) < mcs then
        Merge c to its closest cluster in C;
      EndIf
    EndFor
```

Our algorithm can automatically detect a suitable number of clusters. Unlike traditional clustering algorithms which need convergence loops, our algorithm takes only one loop to identify the clusters, and one loop to merge small clusters.

In K-means algorithm, it randomly selects some data points as the initial center of clusters, and the quality of clustering greatly depends on this selection, especially on event log, where a same trace can occur several times as depicted in Fig. 3, where the trace *acfdh* repeats 10 times. The repeated traces with a big number of times should be a cluster candidate. One more advantage of the algorithm is the ability to put repeated traces into a cluster candidate and removes the uncertainty of random.

The proposed algorithm needs one loop for context tree construction, one loop for clustering. Thus, its complexity is much less than that of traditional clustering algorithms such as K-means, K-modes. Furthermore, the proposed algorithm does not need to transform trace in an intermediate representation (e.g., binary, k-gram, maximal pair, maximal repeat, super maximal repeat and near super maximal repeat, etc.), convert this representation into vector, since it works directly with the traces, then the pre-processing time is greatly reduced.

### 3.5 An Application Framework for ContextTracClus Algorithm

In process discovery application, we propose a framework as described in Fig. 4, which consists of 5 steps.

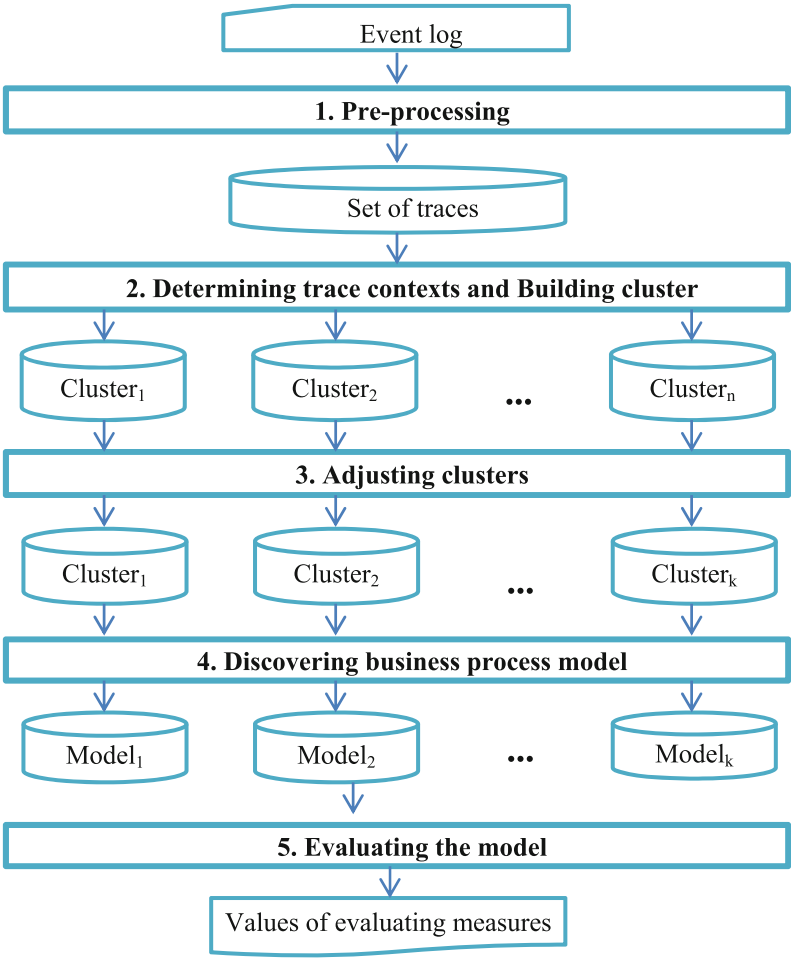


**Fig. 4.** An application framework of the ContextTracClus algorithm

The Pre-processing step transforms the input event log into a list of traces, i.e., merger all the events with the same *caseid* in the event log into a sequence of activities (sorted by recorded time) to form a trace [20, 23].

Step 2 and 3 use ContextTracClus algorithm to determine the contexts that appear in the event log, and generate $n$ clusters. After adjustment, the number of clusters is $k$, where $k \leq n$. Each cluster is used to create a sub-log for process discovery.

In step 4, the α-algorithm is used to generate the sub-process models corresponding to each event sub-log.

The Evaluating model step evaluates the quality of each generated process models by two *Fitness* and *Precision*. The fitness measure determines whether all traces in the log can be replayed by the model from beginning to end. The precision measure determines whether the model has behavior very different from the behavior seen in the event log. Additional explanation about the fitness: consider an event log $L$ of 600 traces, and $M$ is the correspondingly generated model. If only 548 traces in $L$ can be replayed correctly in $M$, then the fitness of $M$ is $\frac{548}{600} = 0.913$. The range of those measures is between 0 and 1, its best value is 1 meaning that the generated process models have the highest quality. Since $k$ models are generated corresponding to $k$ clusters, the final measures, i.e., fitness and precision, are calculated as formula (1).

$$w_{avg} = \sum_{1}^{k} \frac{n_i}{n} w_i \tag{1}$$

where $w_{avg}$ is the aggregated value of the fitness or precision measure, $k$ is the number of clusters, $n$ is the number of traces in the event log, $n_i$ and $w_i$ are the number of traces and the value of the measure in the $i^{th}$ cluster, correspondingly [18].

## 4 Experimental Result Evaluation

To evaluate the effectiveness of the proposed trace clustering algorithm, we compare our proposed algorithm with K-means clustering algorithm, on three different event logs, i.e., Lfull[1], prAm6[2] and prHm6 (see Footnote 2). Lfull includes 1391 cases with 7539 events; prAm6 consists of 1200 cases with 49792 events; and prHm6 contains 1155 cases with 1720 events.

In the experiment with K-means clustering algorithm, the $k$-grams trace representation $(k = 1, 2, 3)$ for binary vectors was used. To generate the process model and evaluate the processes, ProM 6.6[3], a process mining tool, was used. The experimental results are shown in Table 3.

[1] www.processmining.org/event_logs_and_models_used_in_book/Chapter7.zip

[2] http://data.3tu.nl/repository/uuid:44c32783-15d0-4dbd-af8a-78b97be3de49

[3] http://www.processmining.org/prom/start

**Table 3.** Results of K-means and ContextTracClus trace clustering algorithm

| Algorithm | Event log | | | | | |
|---|---|---|---|---|---|---|
| | Lfull | | prAm6 | | prHm6 | |
| | Fitness | Precision | Fitness | Precision | Fitness | Precision |
| *Scenario 1: Using K-means algorithm* | | | | | | |
| 1-g | **0.991** | 0.754 | 0.968 | 0.809 | 0.902 | 0.66 |
| 2-g | 0.951 | 0.958 | 0.968 | 0.809 | 0.902 | 0.66 |
| 3-g | 0.955 | 0.962 | 0.968 | 0.809 | 0.902 | 0.66 |
| *Scenario 2: Using ContextTracClus algorithm* | | | | | | |
| | 0.982 | **1** | **0.975** | **0.904** | **0.922** | **0.673** |

The experimental results show that ContextTracClus always has a higher precision, i.e., it ensures that the generated process model has the least behaviors not seen in the event log. This is because the traces in a cluster have the same context, i.e., they have the same set of actions so the generated model will have at least superfluous behaviors.

In the scenario 1, we found out the most suitable number of clusters for the data set is 3 after trying with different numbers of clusters, such as 2, 3, 4, 5. The scenario 2 automatically detected the number of clusters based on the input size threshold.

## 5 Related Work

Greco et al. [4] proposed a clustering solution on traces in event log using bag-of-activities trace representation for *K*-means algorithm.

Song et al. [11] presented a trace clustering approach based on log profiles which captured the information typically available in event logs e.g., activity profile, originator profile. In their approach, the *K*-means, Quality Threshold, Agglomerative Hierarchical Clustering, and SelfOrganizing Maps clustering algorithms were used.

Jagadeesh Chandra Bose et al. [20] proposed a trace representation method based on using some control-flow context information e.g., Maximal Pair, Maximal Repeat, Super Maximal Repeat and Near Super Maximal Repeat. They used some of the clustering algorithms such as Agglomerative Hierarchical Clustering, *K*-means.

Weerdt et al. [6] proposed the ActiTraC algorithm, a three-phase algorithm for clustering an event log into a collection of sub-logs to increase the quality of the process discovery task. The ActiTraC algorithm includes three phases: Selection, Look ahead, and Residual trace resolution. The important idea of this algorithm is the sampling strategy, i.e., a trace is added to the current cluster if and only if it does not decrease the process model accuracy too much.

Ha et al. [18] provided a trace representation solution based on the distance graph model for *K*-Modes, *K*-means clustering algorithms. In this representation, it can describe the ordering and the relationship between the activities in a trace. Distance graphs order $k$ of a trace describe the activity pairs which has distance at most $k$ activities in the trace.

Baldauf et al. [12] presented a survey on an architecture of context-aware systems, which includes the design principles, the common context models. They introduced the existent context-aware systems and discussed their advantages and disadvantages. Their paper mentioned a number of different definitions of "context" such as location, identities of nearby people, objects and changes to those objects (Schilit and Theimer 1994); The user's location, environment, identity and time (Ryan et al. 1997); The user's emotional state, focus of attention, location and orientation, date and time, as well as objects and people in the user's environment (Dey 1998); The aspects of the current situation (Hull et al. 1997). The elements of the user's environment which the computer knows about (Brown 1996).

Becker et al. [22] introduced the support of context information in analyzing and improving processes in logistics. They defined the context as time, location, and frequency of events, tools, devices, or operators. In the experiments, they used the frequency of a process and its overall cycle time as the context data. In addition, they used K-Medoids clustering algorithm for the identification of process groups and for the evaluation of context information.

Bolt et al. [1] presented an unsupervised technique to detect relevant process variants in event logs by applying existing data mining techniques. This technique splits a set of instances based on dependent and independent attributes.

Leyer [13] presented a new approach to identify the effect of context factors on business process performance in the aspect of processing time. They proposed a two-stage approach to identify the relevant data and to determine the context impact by applying the statistical methods.

## 6 Conclusions and Future Work

This paper proposed a definition of context in business process and a new trace clustering algorithm base on contexts. A context tree was introduced to make the complexity of the algorithm is reduced with two loops over the input for finding clusters, and one small loop over the clusters for adjustment. The ability to work directly with the traces without transforming to an immediate representation is an additional advantage of the algorithm. Another ability to automatically detect the optimal number of clusters makes algorithm to remove the disadvantage of traditional clustering algorithms and produce determined results. As future work, we plan to study the impact of the context in other tasks of the process mining.

## References

1. Bolt, A., van der Aalst, W.M.P., de Leoni, M.: Finding process variants in event logs. In: Panetto, H., et al. (ed.) On the Move to Meaningful Internet Systems. OTM 2017 Conferences. OTM 2017. LNCS, vol. 10573, pp. 45–52. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69462-7_4
2. Dey, A.K.: Context-aware computing: the CyberDeskProject. In: Proceedings of the AAAI, Spring Symposium on Intelligent Environments, pp. 51–54 (1998)

3. Schilit, B.N., Adams, N., Want, R.: Context-aware computing applications. In: WMCSA, pp. 85–90 (1994)
4. Greco, G., Guzzo, A., Pontieri, L., Saccà, D.: Discovering expressive process models by clustering log traces. IEEE Trans. Knowl. Data Eng. **18**, 1010–1027 (2006)
5. Fischer, I., Poland, J.: New methods for spectral clustering. In: Proceedings of ISDIA (2004)
6. Weerdt, J.D., vanden Broucke, S.K.L.M., Vanthienen, J., Baesens, B.: Active trace clustering for improved process discovery. IEEE Trans. Knowl. Data Eng. **25**(12), 2708–2720 (2013)
7. Poland, J., Zeugmann, T.: Clustering the Google distance with eigenvectors and semidefinite programming. Knowl. Media Technol. **21**, 61–69 (2006)
8. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: SIGMOD Conference, pp. 1–12 (2000)
9. Weerdt, J.D.: Business process discovery_new techniques and applications. Runner up Ph.D. thesis (2014)
10. Evermann, J., Thaler, T., Fettke, P.: Clustering traces using sequence alignment. In: Reichert, M., Reijers, Hajo A. (eds.) BPM 2015. LNBIP, vol. 256, pp. 179–190. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42887-1_15
11. Song, M., Günther, Christian W., van der Aalst, Wil M.P.: Trace clustering in process mining. In: Ardagna, D., Mecella, M., Yang, J. (eds.) BPM 2008. LNBIP, vol. 17, pp. 109–120. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00328-8_11
12. Baldauf, M., Dustdar, S., Rosenberg, F.: A survey on context aware systems. IJAHUC **2**(4), 263–277 (2007)
13. Leyer, M.: Towards a context-aware analysis of business process performance. In: PACIS, vol. 108 (2011)
14. Ryan, N., Pascoe, J., Morse, D.: Enhanced reality fieldwork: the context-aware archaeological assistant. In: Proceeding of the 25th Anniversary Computer Applications in Archaeology (1997)
15. Vitányi, P.M.B.: Information distance: new developments. CoRR abs_1201.1221 (2012)
16. De Koninck, P., De Weerdt, J., vanden Broucke, S.K.L.M.: Explaining clusterings of process instances. Data Min. Knowl. Discov. **31**(3), 774–808 (2017)
17. Koninck, P.D., Weerdt, J.D.: Determining the number of trace clusters_a stability-based approach. In: ATAED@Petri Nets_ACSD, pp. 1–15 (2016)
18. Ha, Q.-T., Bui, H.-N., Nguyen, T.-T.: A trace clustering solution based on using the distance graph model. In: Nguyen, N.-T., Manolopoulos, Y., Iliadis, L., Trawiński, B. (eds.) ICCCI 2016. LNCS (LNAI), vol. 9875, pp. 313–322. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45243-2_29
19. Jagadeesh Chandra Bose, R.P., van der Aalst, W.M.P.: Context aware trace clustering: towards improving process mining results. In: SDM 2009, pp. 401–412 (2009)
20. Jagadeesh Chandra Bose, R.P.: Process mining in the large preprocessing, discovery, and diagnostics. Ph.D. thesis, Eindhoven University of Technology (2012)
21. Thaler, T., Ternis, S.F., Fettke, P., Loos, P.: A comparative analysis of process instance cluster techniques. Wirtschaftsinformatik **2015**, 423–437 (2015)
22. Becker, T., Intoyoad, W.: Context aware process mining in logistics. Procedia CIRP **63**, 557–562 (2017)
23. Van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19345-3