UNIVERSITY
OF WOLLONGONG
AUSTRALIA

University of Wollongong
Research Online

Faculty of Engineering and Information Sciences - Papers: Part A

Faculty of Engineering and Information Sciences

2014

# Mining periodic patterns from nested event logs

Janusz R. Getta
*University of Wollongong*, jrg@uow.edu.au

Marcin Zimniak
*Tu Chemnitz, Germany*

Wolfgang Benn
*Tu Chemnitz, Germany*

## Publication Details

Getta, J. R., Zimniak, M. & Benn, W. (2014). Mining periodic patterns from nested event logs. Computer and Information Technology (CIT), 2014 IEEE International Conference on (pp. 160-167). United States: Institute of Electrical and Electronics Engineers.

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au

# Mining periodic patterns from nested event logs

**Abstract**

2014 IEEE. Information about periodic computations of processes, events, and software components can be used to improve performance of software systems. This work investigates mining periodic patterns of events from historical information related to processes, events, and software components. We introduce a concept of a nested event log that generalizes historical information stored in the application traces, event logs and dynamic profiles. We show how a nested event log can be compressed into a reduced event table and later on converted into a workload histogram suitable for mining periodic patterns of events. The paper defines a concept of periodic pattern and its validation in a workload histogram. We propose two algorithms for mining periodic patterns and we define the quality indicators for the patterns found. We show, that a system of operations on periodic patterns introduced in this work can be used to derive new periodic patterns with some of the quality indicators better from the original ones. The paper is concluded with an algorithm for deriving periodic patterns with the given quality constraints.

**Keywords**

patterns, nested, event, mining, logs, periodic

**Disciplines**

Engineering | Science and Technology Studies

**Publication Details**

Getta, J. R., Zimniak, M. & Benn, W. (2014). Mining periodic patterns from nested event logs. Computer and Information Technology (CIT), 2014 IEEE International Conference on (pp. 160-167). United States: Institute of Electrical and Electronics Engineers.

This conference paper is available at Research Online: http://ro.uow.edu.au/eispapers/4061

# Mining Periodic Patterns from Nested Event Logs

Janusz R. Getta
School of Computer Science and Software Engineering
University of Wollongong
Wollongong, Australia
Email: jrg@uow.edu.au

Marcin Zimniak and Wolfgang Benn
Faculty of Computer Science, TU Chemnitz
Chemnitz, Germany
Email: {marcin.zimniak,benn}@cs.tu-chemnitz.de

***Abstract*—Information about periodic computations of processes, events, and software components can be used to improve performance of software systems. This work investigates mining periodic patterns of events from historical information related to processes, events, and software components. We introduce a concept of a nested event log that generalizes historical information stored in the application traces, event logs and dynamic profiles. We show how a nested event log can be compressed into a reduced event table and later on converted into a workload histogram suitable for mining periodic patterns of events. The paper defines a concept of periodic pattern and its validation in a workload histogram. We propose two algorithms for mining periodic patterns and we define the quality indicators for the patterns found. We show, that a system of operations on periodic patterns introduced in this work can be used to derive new periodic patterns with some of the quality indicators better from the original ones. The paper is concluded with an algorithm for deriving periodic patterns with the given quality constraints.**

## I. Introduction

Process mining discovers the structures of real world processes from information recorded in the event logs, traces from performance tests, dynamic profiles, audit trails, etc. Process mining transforms historical information into the formal models of concurrent processes like for example Petri nets, Workflow nets, BPM notation, Event-Driven Process chains and the others. Apart from the main objective of process *discovery*, process mining also contributes to the *conformance checking* and *enhancement* of the real world processes [1].

*Conformance checking* compares an already created process model with the logs of its executions recorded at the testing stages to make sure that unexpected situations do not happen when the process runs at a production stage.

*Enhancement* improves the performance indicators of the existing processes through analysis of historical data merged with performance data. In particular, the objectives of *enhancement* through process mining include efficient handling of workload peaks, detection of performance bottlenecks, increasing process throughput, decreasing response time, and the other performance related problems. For example, a typical application of *process enhancement* is an automated physical database design and performance tuning where information about the periods of exceptionally low and high workload is used by a database system to restructure data, to reduce access paths, to cluster the groups of related data, and all what prepares a database system for more efficent future processing [2].

The workload peaks and longer periods of very high or very low workload are inevitable due to the interferences of periodically performed processes and due to randomly occurring ad-hoc processes reflecting the real world random events. To reduce a negative impact of the workload peaks on the overall performance of a system it should be possible to anticipate the workload peaks and to "prepare" a system through re-scheduling of the less important activities, through creating better access paths to data resources, and through allocation of more computational resources to the complex processes. One of the ways how workload peaks can be anticipated is through the discovery of *periodic processes* performed by the user applications.

To prepare a software system for a high workload peak more information is needed about the periodicity of events and also about the implementation details of events recorded in the logs and other performance related information that can be collected from the dynamic profiles. In particular, we need information about the sequences and frequencies of software components executed during the computations of processes and events. For example, it may happen that processing of several different events requires the periodic computations of the same software components and because of that it creates a clear processing pattern for the components. Such information can be obtained from the associations of user application logs, event logs, with the dynamic profiles of software components implementing a system [3]. Dynamic profiling of software provides time related information about the software components used when processing the events. The results of dynamic profiling can be customized in a number of ways to a collect information about pre-specified components, to set a given level of nested invocations of components, and to filter out information not relevant to the processing.

In this work we adopt a multilevel model of historical information used for process mining. At the topmost level we consider the sequences of processes performed by the user applications. At a lower level, we consider the sequences of events performed by the processes and recorded in the event logs. At the levels below, we consider information about the sequences of executions of nested software components stored in the dynamic profiles. For example, in a relational database systems, the sequences of processes are determined by the database applications implementing the real world business processes. The events are the processing steps of user applications. The software components are SQL statements, and the relational algebra operations on relational tables.

The main objective of this work is to use aggregated information from the application logs, event logs, and internal processing of events included in the dynamic profiles to discover the *periodic patterns* of processes, events, and operations implementing the events.

To solve the problem, we simplify a model of processes, events, and executions of software components. We generalize a concept of *event* to represent processes, events, and operations recorded at the different levels of traces, logs, and dynamic profiles. Next, we define a model of *nested event log* that generalizes the applications traces, event logs, and dynamic profiles. We show, that it is possible to transform the application traces, events logs, and dynamic profiles into a nested event log through comparison of timestamps collected from application traces, event logs and dynamic profiles. A formal view of the results from merging application traces with event logs and dynamic profiles is a sequence of nested events where each one is associated with a time slot when an event has happened and each event is a possibly empty sequence of lower level events.

Data preparation starts from partitioning of a period of time over which a nested event log was recorded into the disjoint time units. The nested events from the application traces, logs, and dynamic profiles are extracted from the logs and stored in an event table which represents the hierarchies of events. Each event in the table is associated with a set of timestamps determining the moments in time when the respective complex or elementary events have been computed. An event table is further reduced by elimination of the events that inherit their properties from their parent events.

In the next stage we use timestamps associated with the events in a reduced table of events and a sequence of disjoint time units to create a workload histogram that contains information about total number of times each event was processed in each one of the assumed time units. The events that have a minimal impact on the overall workload are eliminated from the workload histogram. Finally, the histogram is used by an algorithm that discovers the periodic patterns of events. The operations on periodic patterns allows for transformation of the patterns, like for example, modification of pattern quality indicators, concatenation of patterns, and decomposition of patterns. The transformations of periodic patterns are used to derive new periodic patterns that satisfy the given values of selected quality indicators.

The paper is organized in the following way. The next section reviews the research works related to processes mining, dynamic profiling and discovering periodic patterns. Section III defines a model of nested events, time units, and workload histograms. In Section IV we present how an event table can be created from a sequence of nested events and how it can be reduced through elimination of events that inherit their patterns from the parent events. Section V defines a concept of periodic pattern and its validation in a workload histogram. Two algorithms for discovering periodic patterns are presented in a Section VI. A system of operations on periodic patterns of events is proposed in a Section VII. Section VIII shows how to apply a set of periodic patterns to performance enhancement through predicting the future workload. Finally, Section IX concludes the paper.

## II. Previous work

The book [1] is at the moment the most comprehensive source of information on the present state of process mining. Since its introduction the problem of process mining has been formally defined and the first solutions have been proposed in an number of practical applications. One of the first application of process mining was to workflow mining [4]. The process mining techniques have also been applied to discover the structures of social networks from the event logs [5] and to fraud detection [6]. Application of process mining to conformance checking has been investigated in [7] and [8].

Recently, processing and analysis of the large event logs that include information about many different processes has been identified as one of the centrals problem in process mining. The papers [9] and [10] propose a very general divide-and-conquer approach based on partitioning of activities to deal with the large event logs. A work [11] introduces the new algorithms ($\alpha$-algorithm, state-based regions, and language-based regions) to discover Petri nets from the large event logs.

A website [12] contains the most of up to date information on process mining research, tools, and applications.

An idea of dynamic profiling of software systems to discover the performance bottlenecks has been proposed in [13]. Then, different types of profiles have been invented to collect information about behavior of software systems [14] and such information has been applied to improve data flow [3]. A work [15] shows how to summarize the results from a number of large dynamic profiles. Recently, a unified representation of dynamic profiles has been proposed in [16] and [17]. Technical information on dynamic profiling can be found in [18].

The works on mining frequent itemsets/association rules [19], frequent episodes [20], and its extensions on mining complex episodes [21] inspired the works on cyclic patterns. A starting point to many research studies on discovering cyclic patterns is a work [22] that defines the principle concepts of cycle pruning, cycle skipping, cycle elimination heuristics.

Discovering periodic patterns in event logs appears to be quite similar to periodicity mining in time series [23] and [24] where the long sequences of elementary data items partitioned into a number of ranges and associated with the timestamps are analyzed to find the cyclic trends. However, due to the internal structures of complex data processing operations, like for example SQL statements, its analysis cannot be treated in the same way as analysis of sequences of atomic data items like numbers of characters.

The latest works on discovering periodic patterns address the concepts of full periodicity, partial periodicity, perfect and imperfect periodicity [25] and the most recently asynchronous periodicity [26] and [27]. The works [28] and [29] review a class of data mining techniques based on analysis of ordered set of operations on data performed by the user applications. The model of periodicity considered in this paper is an extension of the model introduced in [30].

## III. Nested event log

A nested event log is a composition of information from a trace of user applications, flat top level log of events and dynamic profile that records behavior of a system at the lower level of individual software components. A trace of user application contains information about the processes performed by the application on behalf of the human operators. An event log contains the sequences of events recorded while the processes were running. A dynamic profile contains information about software components (operations) executed during processing of an event. In this work, we generalize information about the processes, events, and operations into a class of *nested events*. Top level events in a hierarchy of nested events represent processes, the usual events triggered by the processes are included at a level below, and then there is a theoretically unlimited number of levels of operations implemented by the software components. In the model a log of an event $e$ is a possible empty sequence of events $\langle e_1, \ldots, e_n\rangle$ such that the same event can be repeated in the sequence many times. The operations that do not invoke any other operations are located at the leaf levels in a hierarchy of nested events.

A *time unit* $t$ is a pair $\langle s, \tau\rangle$ where $s$ is a start point in time of the unit and $\tau$ is its length.

A *nested log* of an event $e$ recorded in a time unit $t_e$ is denoted by $L_{t_e}(e)$ and it is a finite and possibly empty sequence of triples $\langle e_1{:}t_1{:}L_{t_1}(e_1), \ldots, e_n{:}t_n{:}L_{t_n}(e_n)\rangle$ where each $e_i$ is a unique identifier of an event, $t_i$ is a time unit when an event $e_i$ occurred, and $L_{t_i}(e_i)$ is a nested log of event $e_i$ recorded in a time unit $t_i$. All time units in a log $L_{t_e}(e)$ are pairwise disjoint and they are used to identify the nested logs $L_{t_1}(e_1), \ldots, L_{t_n}(e_n)$. An empty log is denoted by $\emptyset$.

As a simple example consider the following nested logs of event $e_1$ that occurred at a time unit $t_1$ and event $e_2$ that occurred at the time units $t_2$ and $t_3$: $L_{t_1}(e_1)=\langle e_{11}{:}t_4{:}\emptyset, e_{12}{:}t_5{:}L_{t_5}(e_{12}), e_{12}{:}t_6{:}L_{t_6}(e_{12})\rangle$, $L_{t_2}(e_2)=\langle e_{11}{:}t_7{:}\emptyset\rangle$, $L_{t_3}(e_2)=\langle e_{11}{:}t_8{:}\emptyset, e_{13}{:}t_9{:}\emptyset\rangle$, $L_{t_5}(e_{12})=\langle e_{121}{:}t_{10}{:}\emptyset\rangle$, $L_{t_6}(e_{12})=\langle e_{121}{:}t_{11}{:}\emptyset\rangle$. A nested log is flat if all events included in the log have their logs empty. For example the logs $L_{t_3}(e_2)$, $L_{t_4}(e_2)$, $L_{t_5}(e_{12})$, and $L_{t_6}(e_{12})$ are flat. The internal structures of events together with respective time units are visualized in a Figure 1.

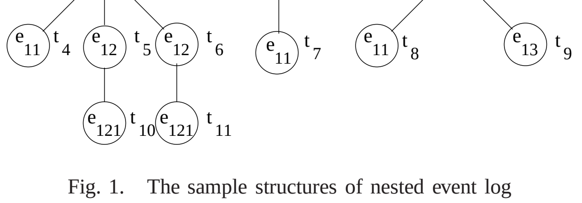

Fig. 1. The sample structures of nested event log

The nested event logs recorded over the long periods of time may consume a lot of space. However, some of the events always have the same internal structure, for example processing of the same SQL statement in different applications accessing the same relational tables is usually performed in the same way. In order to efficiently analyze information included in a *nested event log* we transform it into a more compact *event table* and later on we reduce the table to eliminate events that have the same properties.

A definition of an event table is based on a concept of *multiset*. A multiset $M$ is a pair $\langle S, f\rangle$ where $S$ is a set of values and $f : S \rightarrow N^+$ is a function that determines multiplicity of each element in $S$ and $N^+$ is a set of positive integers [31]. In the rest of this paper we shall denote a multiset $\langle\{e_1, \ldots, e_m\}, f\rangle$ where $f(e_i) = k_i$ for $i = 1, \ldots, m$ as $(e_1^{k_1}, \ldots, e_m^{k_m})$. We shall denote an empty multiset $\langle\emptyset, f\rangle$ as $\emptyset$. We shall abbreviate a single element multiset $(e^k)$ as $e^k$.

A *signature of an event* $e$ in a nested log $L_{t_e}(e)$ is denoted by $S(L_{t_e}(e))$ and it is defined as a multiset of events included in a nested log of an event $e$. For instance, $S(L_{t_1}(e_1)) = (e_{11}, e_{12}^2)$ in the example above. It is important to note, that two signatures of the same event $e$ recorded in the same or different logs can be different due to the constraints imposed in the different periods of time. For example, the invocations of polymorphic methods may return different signatures, or the computations of `SELECT` statement may return different signatures depending on the parameters of a query optimizer. In the example above, the signatures of an event $e_2$ in the logs $L_{t_2}(e_2)$ and $L_{t_3}(e_2)$ are different, $S(L_{t_2}(e_2)) = (e_{11})$ and $S(L_{t_3}(e_2)) = (e_{11}, e_{13})$.

The events $e_1, \ldots, e_n$ whose nested logs have in their signatures an event $e$ are called as *parent events* of an event $e$, For example, an event $e_{11}$ has the following multiset of parent events $(e_1, e_2^2)$.

An *event table* for a nested log $L$ is a set of triples $<e, T, S(e)>$ where $e$ is an event, $T$ is a set of time units when the event had occurred, and $S(e)$ is a set of signatures of the event. If all signatures of an event which occured many times are identical then such event is represented only once in an event table. An event table for a nested event log given in the example above is included in a Table I. An event table

TABLE I
An event table

| Event | Time units | Signature |
|---|---|---|
| $e_1$ | $\{t_1\}$ | $(e_{11}, e_{12}^2)$ |
| $e_2$ | $\{t_2\}$ | $(e_{11})$ |
| $e_2$ | $\{t_3\}$ | $(e_{11}, e_{13})$ |
| $e_{11}$ | $\{t_4, t_7, t_8\}$ | $\emptyset$ |
| $e_{12}$ | $\{t_5, t_6\}$ | $(e_{121})$ |
| $e_{121}$ | $\{t_{10}, t_{11}\}$ | $\emptyset$ |
| $e_{13}$ | $\{t_9\}$ | $\emptyset$ |

contains information about the time units when each event from a collection of nested event logs has been processed and about the internal structures of events, i.e. what events occurred during a processing of each event and additionally how many times each event has been processed. If a multiset of parent events of an event $e_i$ contains only one event $(e_j^k)$ and a value of $k$ is equal to the total number of times and event $e_j$ has been recorded in a nested log then it means that all properties discovered for an event $e_i$ are the same as for an event $e_j$. For example, in an event table above, a multiset of parents events of an event $e_{12}$ is equal to $(e_1)$ and only one event $e_1$ has been recorded in a log. It means, that an event $e_{12}$ inherits all properties of an event $e_1$. On the other hand, if a multiset of parent events of an event $e_i$ contains many instances of the same or different events then its properties are the compositions of properties inherited from the parent events. Of course, it may also happen that an event that has multiple parent events has the new properties not possessed by the parent events.

Therefore, an event table can be further reduced through elimination of events that occur in every instance of their parent events and such that they have only one parent event. In the example above, the events $e_{12}$, $e_{121}$ occur in all instances of one parent event and because of that they can be removed from the event table given in a Table I. An event $e_{11}$ has two parents $(e_1, e_2^2)$ and it cannot be removed. An event $e_{13}$ cannot be removed because it has one parent $(e_2)$ and does not occur in all instances of its single parent because an event $e_2$ has two instances in a nested log. A reduced event table is given in a Table II.

TABLE II
A REDUCED EVENT TABLE

| Event | Time units | Signature |
|---|---|---|
| $e_1$ | $\{t_1\}$ | $(e_{11})$ |
| $e_2$ | $\{t_2\}$ | $(e_{11})$ |
| $e_2$ | $\{t_3\}$ | $(e_{11}, e_{13})$ |
| $e_{11}$ | $\{t_4, t_7, t_8\}$ | $\emptyset$ |
| $e_{13}$ | $\{t_9\}$ | $\emptyset$ |

## IV. WORKLOAD HISTOGRAMS

A model of *workload histograms* described in this section is based on the model developed earlier for audit trails in database systems [30]. We consider a period of time $\langle t_{start}, t_{end}\rangle$ over which a log event and respective dynamic profile are recorded. The period of time is divided into a contiguous sequence of disjoint and fixed size *elementary time units* $\langle t_e^{(i)}, \tau_e\rangle$ where $t_e^{(i)}$ for $i = 1, \ldots, n$ is a timestamp when an elementary time unit starts and $\tau_e$ is a length of the unit. The period $<t_{start}, t_{end}>$ consists of elementary time units such that $t_{start} = t_e^{(1)}$ and $t_e^{(i+1)} = t_e^{(i)} + \tau_e$ and $t_e^{(n)} + \tau_e = t_{end}$.

A time unit $\langle t, \tau\rangle$ consists of one or more consecutive elementary time units. A nonempty sequence $U$ of $n$ disjoint time units $<t^{(i)}, \tau^{(i)}>$ $i = 1, \ldots, n$ over $\langle t_{start}, t_{end}\rangle$ is any sequence of time units that satisfies the following properties: $t_{start} \leq t^{(1)}$ and $t^{(i)} + \tau^{(i)} \leq t^{(i+1)}$ and $t^{(n)} + \tau_{(n)} \leq t_{end}$.

As a simple example consider a nested event log that starts on $t_{01:01:2007:0:00am}$ and ends on $t_{31:01:2007:12:00pm}$. Then, a sequence of disjoint time units called as *morning tea time* consists of the following units $\langle t_{01:01:2007:10:30am}, 30\rangle$, $\langle t_{02:01:2007:10:30am}, 30\rangle$, ..., $\langle t_{31:01:2007:10:30am}, 30\rangle$.

Let $|U|$ denotes the total number of time units in $U$ and let $U[n]$ denotes the $n$-th time unit in $U$ where $n$ changes from 1 to $|U|$. A *workload histogram of an event* $e$ is a sequence $W_e$ of $|U|$ multisets of events such that $W_e[i] = (e^{f_i})$ and $f_i \geq 1$ is equal to the total number of times an event $e$ has been processed in the $i$-th time unit $U[i]$.

If a time unit of an event $e$ overlaps on more than one time unit in $U$ then the event still contributes to only one element in a workload histogram. Such event is included in an element of a histogram where it spent the majority of time and if more than one such time unit in $U$ exists then a time unit in $U$ with the smallest index is selected. A workload histogram of an event $e$ is created from information about time units in $U$ and the values of time units for the event in a reduced event table.

Let $E$ be a set of all events obtained from a nested event log $L$ and recorded in a reduced event table. A *workload histogram of a nested event log* $L$ is denoted by $W_L$ and $W_L[i] = \biguplus_{e \in E} W_e[i], \forall i = 1, \ldots, |U|$, i.e. it is a sum of workload histograms of all events included in a reduced event table.

## V. PERIODIC PATTERNS

A *periodic pattern* is a triple $\langle e^k,\ f{:}t,\ n{:}x\rangle$ where: $e^k$ is a multiset of event $e$, $f{:}t$, is a range of $\underline{f}rom$ and $\underline{t}o$ values, such that $1 \leq f < t \leq |U|$, and $n{:}x$ is a range of $mi\underline{n}$ and $ma\underline{x}$ values such that $n \leq x$ and $(t - f + 1) \geq x$. If $f = 1$ and $t = |U|$ then $f{:}t$ of is abbreviated to $|U|$. If $n = x = p$ then $n{:}x$ is abbreviated to $p$.

Let $W_L$ be a workload histogram of a nested log $L$. We say, that a periodic pattern $\langle e^k,\ f{:}t,\ n{:}x\rangle$ *is valid in a workload histogram* $W_L$ if there exists a sequence $\langle W_L[i_1], \ldots, W_L[i_m]\rangle$ such that:
(i) $e^{k_j} \subseteq W_L[i_j]$ such that $k \leq k_j$ for $j = 1, \ldots, m$ and
(ii) $i_1 = f$ and $i_m = t$ and
(iii) $i_{j+1} = i_j + n$ or $i_{j+1} = i_j + n + 1$ or ... or $i_{j+1} = i_j + x$ for $j = 1, \ldots, m - 1$.

For example, the parameters of a periodic pattern $\langle e^2$, 1:10, 2:3$\rangle$ mean that a workload histogram $W_L$ contains a sequence of elements such that an event $e$ is repeated at least two times in $W_L[1]$ and it is repeated at least two times in $W_L[10]$ and the repetitions of $e$ also occurs at least two times in the elements of the sequence which are distant from each other by no less than 2 elements and no more than 3 elements. We assume that a distance between adjacent elements in $W_L$ is equal to 1. In the other words, there exists a sequence in $W_L$ whose elements at least 1 element apart and at most 2 elements of $W_L$ apart and such that the repetitions of $e$ occur at least two times in all elements of the sequence.

In another example, a periodic pattern $\langle e_1^2$, 2:6, 1:3$\rangle$ is valid in a workload histogram $W_L = \langle\emptyset$, $(e_1^2, e_2)$, $e_2^2$, $e_2^3$, $\emptyset$, $(e_1^3, e_2)$, $e_2^2\rangle$ because $e_1^2 \subseteq W_L[2]$ and $e_1^2 \subseteq W_L[6]$ and for each element $W_L[i]$ that includes $e_1^2$ the next element $W_L[j]$ that also includes $e_1^2$ satisfies a condition $i + 1 \leq j \leq i + 3$.

In a special case the parameters of a periodic pattern $\langle e, |U|, 2\rangle$ mean that an event $e$ is repeated at least one time in every second element of $W_L$ starting from the first element of workload histogram and ending in the last one.

The model of periodic pattern defined above is general enough to represent two main types of periodic repetitions of events. The first one, when an event is repeated a given number of times in every given period of time. For example, a periodic pattern in which an event $e$ is repeated three times every second day can be represented by choosing days as time units and partitioning workload by days, assuming a multiset of events $e^3$, and assuming that $n = x = 2$. In the second type of periodic repetitions an event is repeated in a given and variable period of time after the previous occurence of the same event. For example, an event $e$ is repeated in a period of time from 5 to 10 minutes after the previous event can be represented by choosing minutes as time units, assuming a multiset $e$ of events, and making the parameters $n$:$x$ equal to 5:10.

A *single event pattern* is a pair $\langle e^k, f\rangle$ where $e^k$ is a multiset of event $e$ and $f$ is a location in a workload histogram $W_L$ such that $e^k \subseteq W_L[f]$. Every periodic pattern is composed from at least two single event patterns. However, a single event pattern does not provide any information about periodicity of events and it is mainly needed to simplify some of the operations on periodic patterns.

Quality of periodic patterns discovered from the nested event logs can be measured in a number of different ways. Let $\langle e^k, f{:}t, n{:}x\rangle$ be a periodic pattern valid in $W_L$. Then, a *length* of periodic pattern is defined as the total number of elements in $W_L$ that validate the pattern. A *length* of periodic pattern defined in this way varies from $\frac{t-f+1}{x}$ to $\frac{t-f+1}{n}$.

Another important quality indicator is *regularity* of periodic pattern defined as the inverse of a difference between the parameters $n$ and $x$. *Regularity* is a fraction in a range (0,1] and it is equal to $\frac{1}{x-n+1}$. *Regularity* of the patterns where $n = x$, i.e. the patterns that have a constant period of repetition of $e^k$ is the highest and it is equal to 1.

Yet another quality indicator related to a distribution of multiset of events $e^k$ in a workload histogram is a *density* of periodic pattern defined as inverse of the largest possible distance between the elements of a sequence in $W_L$ that validates the pattern. *Density* is a fraction in a range (0,1] and it is equal to $\frac{1}{x+n-1}$. The pattern that have $n = x = 1$ have the highest density equal to 1 because every element of workload histogram in a range from $f$ to $t$ contains $e^k$.

The last quality parameter is a *weight* of periodic pattern defined as the total number of repetitions $k$ of event $e$ in the pattern. *Weight* of a periodic pattern $\langle e^k, f{:}t, n{:}x\rangle$ is equal to $k$.

When it comes to an overall evaluation of the quality we say that a high quality periodic pattern is long, regular, dense, and it has a high weight.

## VI. Mining periodic patterns

Input to the algorithms mining periodic patterns is a nested log of events $L$ created by associating and merging the application traces, event logs and dynamic profiles. At a data preparation stage, the log is first converted into an event table and later on into a reduced event table. A reduced event table and a sequence of time units $U$ are used to create a workload histogram $W_L$.

The simplest way to find a periodic pattern in a nested log of event is to consider the patterns based on a given multiset of events $e^k$. We say that two multisets $e^{k'}$ and $e^{k''}$ are *adjacent* in a workload histogram if $e^{k'} \subseteq W_L[i]$ such that $k \leq k'$ and $e^{k''} \subseteq W_L[j]$ such that $k'' \leq k$ and $\neg\exists\ i<n<j$ such that $e^{k'''} \subseteq W_L[n]$ such that $k''' \leq k$. A *distance* between two adjacent multisets $e^k$ in a workload histogram $W_L$ is equal to $|i - j|$.

A *trace* of a multiset $e^k$ in a workload histogram $W_L$ is a sequence of positive integer numbers $\langle n_1, \ldots, n_m\rangle$ such that each $n_i$ is equal to a distance between two adjacent multisets $e^k$ in a workload histogram $W_L$ for $i = 1, \ldots, m$. For example, a trace of a multiset $e^2$ in a workload histogram $W_L = \langle e^2, e, e^3, e^2, e^4, \emptyset, e^5, e, e^2, e^3\rangle$ is a sequence of numbers $\langle 2, 1, 1, 2, 2, 1\rangle$.

The following property provides a justification for a simple algorithm for mining periodic patterns in a workload histogram $W_L$. A periodic pattern $\langle e^k, f{:}t, n{:}x\rangle$ is valid in $W_L$ if $e^{k'} \subseteq W_L[f]$ such that $k \leq k'$ and $e^{k''} \subseteq W_L[t]$ such that $k \leq k''$and $n \leq min(T)$ and $x \geq max(T)$ where $T$ is a trace of a multiset $e^k$ in sub-sequence of workload histogram that starts in $W_L[f]$ and ends at $W_L[t]$. An algorithm below uses the property for mining a periodic pattern $\langle e^k, f{:}t, n{:}x\rangle$.

**Algorithm 1**

```
(1)   We iterate over the elements of a workload histogram $W_L$
      from the first to the last element.
(1.1) We record in $f$ an index of the first element in a workload
      histogram such that $e^{k'} \subseteq W_L[f]$ where $k \leq k'$ and we
      record in $t$ an index of the last element $e^{k''} \subseteq W_L[t]$
      where $k \leq k''$.
(1.2) We find a trace $T$ of a multiset $e^k$ in a workload histogram
      $W_L$.
(1.3) We find $n := min(T)$ and $x := max(T)$.
(2)   We output a periodic pattern $\langle e^k, f{:}t, n{:}x\rangle$.
(3)   We modify the entries in a workload histogram $W_L$
      in the following way $W_L[f] := W_L[f] - e^k$, $W_L[f + t_1] := W_L[f + t_1] - e^k$,...,$W_L[f] := W_L[f] - e^k$ where
      $t_1, \ldots, t_k$ are the elements of a trace $T$. Such modifica-
      tion is needed to eliminate an impact of a periodic pattern
      $\langle e^k, f{:}t, n{:}x\rangle$ on a workload histogram $W_L$.
```

Apart from a weight $k$ of periodic pattern, the algorithm above does not impose any limitations on the values of parameters $f$, $t$, $n$, and $x$ and in the consequence any minimal requirements on the quality indicators of the patterns. As the algorithm requires one discovery and one modification pass through a workload histogram its complexity is $O(2h)$ where $h$ is the total number of elements in a workload histogram.

The next algorithm imposes more control on the regularity and density of the discovered patterns. Let $V$ be a multiset and let $e^{k'} \subseteq V$. The result of a difference of multisets $V - (e^k)$ where $k \leq k'$ is a multiset $V'$ that contains all elements from $V$ except an element $e^{k'}$, which is changed to $e^{k'-k}$ and if $k' \leq k$ then an element $e^{k'}$ is not included in $V'$.

**Algorithm 2**

(1) We initialize the values of $x$ and $n$ to achieve the required values of regularity and density. In the outermost loop we iterate over all events included in a reduced event table. If no more events are available in a reduced event table then we end the algorithm else we make $e_c$ the current event.

(1.2) Next, we create a histogram $W_{e_c}$ for the current event $e_c$ and we find a set $K$ of repetitions of $e_c$ in a workload histogram $W_L$.

(1.3) In the next step we iterate over the values in a set $K$ starting from the smallest value $k_c \in K$. If no more values can be found in $K$ we return to a step (1). We set a value of a variable $start$ to 1.

(1.3.1) We find the smallest values $f_c$, $t_c \geq start$ such that $e_c^{k'} \subseteq W_{e_c}[f_c]$ and $k_c \leq k'$ and $e_c^{k''} \subseteq W_{e_c}[t_c]$ and $k_c \leq k''$ and $(f_c + n) \leq t_c \leq (f_c + x)$.

(1.3.2) If the values of $f_c$ and $t_c$ cannot be found we remove $k_c$ from $K$ and we return to a step (1.3)

(1.3.3) If the values of $f_c$ and $t_c$ can be found we modify the entries in a workload histogram of an event $e_c$ in the following way: $W_{e_c}[f_c] := W_{e_c}[f_c] - e_c^{k_c}$ and $W_{e_c}[t_c] := W_{e_c}[t_c] - e_c^{k_c}$.

(1.3.4) Next, we check if it is possible to find the smallest value $t'$ such that $t_c + n \leq t' \leq t_c + x$ and $e_c^{k'} \in W_{e_c}[t']$ and $k \leq k'$.

(1.3.5) If $t'$ is found then we set $t_c := t'$ and modify $W_{e_c}[t_c] := W_{e_c}[t_c] - e_c^{k_c}$ and we repeat a step (1.3.4).

(1.3.6) If $t'$ is not found then we output a periodic pattern $\langle e_c^{k_c}$, $f_c$:$t_c$, $n$:$x\rangle$ and we set $start := t_c + 1$.

(1.3.7) If $start + x \leq |U|$ then we repeat a step (1.3.1) else we modify all remaining values $k \in K$ such that $k := k - k_c$ and we return to a step (1.3).

The algorithm passes through a reduced event table and creates a workload histogram for each event in the table. Each histogram is passed one time in a discovery phase and one time in a modification phase. Complexity of the algorithm depends on the total number of events $n_e$ and an average size $h$ of histograms for all events and it is $O(2n_eh)$. The algorithm above clearly favors *length* of the patterns over *weight* of the patterns because the iteration over multisets of events always start from the single instances of events. When mining a pattern over a multiset $e^k$ a value of parameter $n$ must be equal or greater than $min(T)$ where $T$ is a trace of event $e_k$ in a workload histogram. Then, a value of parameter $x$ depends on what values of quality indicators are expected from the discovered patterns. Theoretically, it is possible to extend the algorithm above with two additional iterations over the values of parameters $n$ and $x$. However, such completely blind extensions of the algorithm would provide us with the periodic patterns which can be logically derived from the periodic patterns that have been already found. For example, if a periodic pattern $\langle e^k$, $f$:$t$, $n$:$x\rangle$ is valid in a workload histogram then a periodic pattern $\langle e^k$, $f$:$t$, $n$:$x$+1$\rangle$ is also valid in the same workload histogram. It is more effective to compute the new periodic patterns through application of the operations on periodic patterns found by the Algorithm 2.

## VII. Operations on periodic patterns

We start from the operations that change the parameters of periodic patterns. A $\kappa$ operation on a periodic pattern reduces a weight of a pattern. It is defined such that $\kappa(\langle e^k$, $f$:$t$, $n$:$x\rangle, k')$ = $\langle e^{k'}$, $f$:$t$, $n$:$x\rangle$ where $k' \leq k$. The correctness of the operation is based on a simple observation that an event $e$ repeated $k$ is also repeated $k' \leq k$ times. For example, $\kappa(\langle e^2$, 2:6, 1:3$\rangle$, 1$)$ returns a periodic pattern $\langle e$, 2:6, 1:3$\rangle$.

A $\chi$ operation changes the values of positional parameters $n$, and $x$. It is defined such that $\chi(\langle e^k$, $f$:$t$, $n$:$x\rangle$, $n'$:$x')$ = $\langle e^k$, $f$:$t$, $n'$:$x'\rangle$ where $1 \leq n' \leq n$ and $|U| \geq x' \geq x$. Correctness of the operation can be directly proved from the definition of validation of periodic pattern in a workload histogram. For example, $\chi(\langle e^2$, 2:6, 1:3$\rangle$, 1:4$)$ returns a periodic pattern $\langle e^2$, 2:6, 1:4$\rangle$.

It is possible to expand a range of a periodic pattern, i.e. to decrease $f$ and to increase $t$ parameters through a concatenation operation (+) of two periodic patterns or a periodic pattern and a single event pattern. A concatenation of the periodic patterns that do not overlap $\langle e^k$, $f$:$t$, $n$:$x\rangle$ + $\langle e^k$, $f'$:$t'$, $n'$:$x'\rangle$ = $\langle e^k$, $f$:$t'$, $min(n, n', (f' - t))$:$max(x, x', (f' - t))\rangle$ where $t \leq f'$. An operation $\kappa$ can be used to equal the weight of the arguments before the concatenation. The correctness of the operation can be proved by the application of $\chi$ operation to one or both periodic patterns to get the same values of $n$:$x$ parameters and later on by the application of a gap $f' - t$ between the patterns to adjust the values of parameters $n$:$x$ in the result of concatenation. For example, $\langle e^2$, 2:6, 1:3$\rangle$ + $\langle e^2$, 8:15, 1:2$\rangle$ is equal to a periodic pattern $\langle e^2$, 2:15, 1:3$\rangle$.

If the periodic patterns overlap, i.e. $f' \leq t$ then the result of concatenation is equal to $\langle e^k$, $f$:$t'$, $min(n, n')$:$max(x, x')\rangle$.

Concatenation of a periodic pattern with a single event pattern can be defined in a similar way through the reduction of a pattern $\langle e^k$, $f'$:$t'$, $n'$:$x'\rangle$ such that $t \leq f'$ into a single event pattern $\langle e^k$,$f'\rangle$ and assumption $n' = \infty$ and $x' = 0$. The result of $\langle e^2$, 2:15, 1:3$\rangle$ + $\langle e^2$,19$\rangle$ is a periodic pattern $\langle e^2$, 2:19, 1:4$\rangle$.

The operations defined above can be used to derive the new periodic patterns that satisfy given quality indicators from a given set of periodic patterns. An algorithm, that derives the periodic patterns takes on input a set of periodic patterns $\mathcal{P}$ and the threshold values of some or all quality indicators such as *weight* $(w_t)$, *length* $(f$:$t)_t$, *regularity* $r_t$, and *density* $d_t$. Some of the quality indicators can be skipped, for example it is possible to evaluate the periodic patterns against the length and regularity indicators only. However, a threshold value for at least one quality indicator must be provided.

**Algorithm 3**

(1) We remove from an input set of periodic patterns $P$ all periodic patterns with *weight* less than $w_t$ and we save the results in a set $P_1$.
(2) We find a set of periodic patterns $P_2 = \{k(p, w_t) : p \in P_1\}$ with the same weight equal to $w_t$.
(3) We remove from a set $P_2$ all periodic patterns included in at least one other periodic pattern in $P_2$ and we save the results in a set $P_3$.
(4) We create $n \times n$ matrix $Q$ of quality indicators such $n = |P_3|$ and such that each periodic patterns in $P_3$ is attached to one row and one column.
(5) Let $p_i$ and $p_j$ be the periodic patterns attached to $i$-th row and $j$-th column of a matrix $Q$. We compute $p_{ij} = p_i + p_j$ for all $i, j = 1, \ldots, |P_3|$ and $i \neq j$ and we save in $Q[i, j]$ some or all quality indicators of a periodic pattern $p_{ij}$, such as *weight* $(w_{ij})$, *length* $(f{:}t)_{ij}$, *regularity* $r_{ij}$, and *density* $d_{ij}$.
(6) We find an entry $Q[i, j]$ with the best values of quality indicators accordingly to our measures. The measurement of quality can be implemented as a user defined function that operates on a given set of quality indicators and it returns a number as a value of an overall quality of periodic pattern. If more than one entry in a matrix $Q$ has the same highest values of quality indicators we randomly pick one of them. If none of the entries in $Q$ satisfies the threshold values of quality indicators we end the algorithm.
(7) We remove the rows $i$, $j$ and the columns $i$, $j$ from a matrix $Q$ and we add a new row and a new column both with a periodic pattern $p_{ij}$ attached.
(8) We recalculate the values of quality indicators for the results of concatenation of a periodic pattern $p_{ij}$ and the remaining patterns. We save the results in a matrix $Q$ and we return to a step (6).

## VIII. Predicting future workload

The main objective of mining periodic patterns is to use information abour the regular repetitions of events to predict the future workloads. Practically, the problem reduces to finding a probability of processing an event $e$ in a given time unit $U[j]$. We start from two simple examples that provide the necessary intuitions on how to use a periodic pattern to estimate the chances for processing of an event $e$ in a given time unit $U[j]$.

First, we consider a periodic pattern $\langle e$, 1:20, 3:5$\rangle$ and we find a probability of processing an event $e$ in a time unit $U[8]$. The traces which bring the processing of an event $e$ close to a time unit $U[8]$ include $\langle 3, 3\rangle$, which ends in a time unit $U[7]$ and $\langle 3, 5\rangle$, $\langle 5, 3\rangle$, which both end in a time unit $U[9]$. Therefore, none of the traces "passes through" a time unit $U[8]$ and probability that a given periodic pattern will contribute to the processing of an event $e$ in a time unit $U[8]$ is equal to zero.

In the second example, we consider a periodic pattern $\langle e$, 1:20, 2:3$\rangle$ and we find a probability that the pattern contributes to the processing of an event $e$ in a time unit $U[7]$. The following two unique traces lead to the processing of an event $e$ in a time unit $U[7]$: $\langle 2, 2, 2\rangle$ and $\langle 3, 3\rangle$. Next, we find the unique traces that pass through the time units $U[8]$ or $U[9]$ and not $U[7]$. There are three traces that lead to the processing of an event $e$ in a time unit $U[8]$: $\langle 2, 2, 3\rangle$, $\langle 2, 3, 2\rangle$, $\langle 3, 2, 2\rangle$, and two traces that lead to processing of an event $e$ in a time unit $U[9]$: $\langle 3, 2, 3\rangle$, $\langle 2, 3, 3\rangle$ and such that none of these traces passes through a time unit $U[7]$. All other traces leading to the processing of an event $e$ in the time units $U[10]$, $U[11]$, $\ldots$, $U[20]$ are the extensions of 7 unique traces already found. Therefore, with two traces out of seven passing through a time unit $U[7]$ a probability of processing $e$ in a time unit $U[7]$ is equal to $\frac{2}{7}$.

We consider a periodic pattern $\langle e^k$, $f{:}t$, $n{:}x\rangle$ and a time unit $U[j]$ such that $f \leq j \leq t$. To find a probability of processing an event $e^k$ in a time unit $U[j]$ we have to find the total number of traces $tot_j$ that pass through a time unit $U[j]$ and the total number of traces $tot_{j'}$ that pass through the time units $U[j+1], U[j+2], \ldots, U[j+x-1]$ and such that none of them passes through a time unit $U[j]$. To find all traces $\langle t_1, \ldots, t_m\rangle$ that pass through a time unit $U[j]$ we have to find all sequences of numbers such that $f + t_1 + \ldots + t_m = j$ and $n \leq t_i \leq x$ for all $i = 1, \ldots, m$. In the other words, we have to find all traces that pass through a time unit $U[j]$ for the values of $m$ equal to $\lfloor \frac{j-f}{n} \rfloor$, $\lfloor \frac{j-f}{n} \rfloor - 1, \ldots, \lfloor \frac{j-x}{n} \rfloor$.

To find the values of $t_1, \ldots, t_m$ that satisfy an equation $f + t_1 + \ldots + t_m = j$ for a given $m$ we use a the following $m$ nested loops.

```
$count_m := 0$;
for $t_1 \in \{n, n+1, ..., x\}$
...        ...        ...
  for $t_m \in \{n, n+1, ..., x\}$
    if $(f + t_1 + ... + t_m) = j$ then $count_m$++;
```

Then to find the total number of traces $tot_j$ that end at time unit $U[j]$ we compute $tot_j = count_{\lfloor \frac{j-f}{n} \rfloor} + \ldots + count_{\lfloor \frac{j-x}{n} \rfloor}$.

In almost the same way we can find the total number of traces $tot_{j'}$ that pass through the time units $U[j+1], U[j+2], \ldots, U[j+x-1]$ and such that none of them passes through a time unit $U[j]$. For example, to find the total number of traces $tot_{j+1}$ that pass through a time unit $U[j+1]$ we use the same method as above and we remove all traces that in the same moment pass through a time unit $U[j]$. Then, $tot_{j'} = tot_{j+1} + \ldots + tot_{j+x-1}$ and a probability of processing an event $e^k$ triggered by a periodic pattern $\langle e^k$, $f{:}t$, $n{:}x\rangle$ in a time unit $j$ is equal to $\frac{tot_j}{tot_j + tot_{j'}}$.

## IX. Summary and future work

This paper shows how to discover the periodic patterns of events from information recorded in the nested event logs. We generalize the application traces, event logs and dynamic profiles into the nested event logs and we compress such logs into the reduced event tables. Next, we show how to define a sequence of time units and how to construct a workload histogram from a given sequence of time units and reduced event table. Then we define a concept of periodic pattern and validation of period pattern in a workload histogram. Two algorithms are provided to discover the periodic patterns in the workload histograms. The paper defines several quality indicators for periodic patterns and it shows how the operations on periodic patterns chabge the values of quality indicators. Finally, we show how a set of periodic patterns can be used for the estimation of future workload.

The periodicity of events provides very useful information for *enhancement* of complex processes through the significant improvements of their performance indicators. A set of periodic patterns can be used anticipate the periods of high or low workload time in order to prepare a software system during the periods of relatively low workload time for the periods of high workload time. For example, during low workload time it is possible to reorganize the structures of a database, increase priority of important processes, transfer from remotes sites data which will be needed in the future, perform pre-computations of frequently repeated processes, etc.

The present model of periodic patterns considers only cyclic repetition of an event in a given sequence of time units. It frequently happens that events are related to each other and that periodicity can be extended on the sequences of dependent events. Another interesting extension is related to the best choice of time units over the workload histograms are created. At the moment, with a completely arbitrarily choice of time units may result with either too fine or to coarse granulation of time and in a consequence it may result with a distorted view of periodic patterns. Too long time units will results with the continuous periodic patterns where every element of workload histogram is included in a pattern. Too short time units will provide periodic patterns with low level of quality indicators such as regularity and density. A mechanism is needed to find the most appropriate granulation of time for the parameters of a given event log.

## REFERENCES

[1] W. M. P. van der Alst, *Process Mining Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.

[2] N. Bruno, Ed., *Automated Physical Database Design and Tuning*. CRC Press Taylor and Francis Group, 2011.

[3] G. Ammons and J. R. Larus, "Improving data flow analysis with path profiles," in *Proceedings of the SIGPLAN Conference on Programming Language Design and Implementation.*, 1998.

[4] W. M. P. van der Alst, B. Dongen, J. van Herbst, L. Maruster, G. Schimm, and A. Weijters, "Workflow mining: A survey of issues and approaches," *Data and Knowledge Engineering*, vol. 47, no. 2, 2003.

[5] W. M. P. van der Alst, H. Reijers, and M. Song, "Discovering social networks from event logs," *Computer Supported Cooperative Work*, vol. 14, no. 6, 2005.

[6] M. Jans, J. M. van der Werf, N. Lybaert, and K. Vanhoof, "A business process mining application for internal transaction fraud mitigation," *Expert Systems with Applications*, vol. 38, no. 10, 2011.

[7] M. de Leoni and W. M. P. van der Aalst, "Aligning event logs and process models for multi-perspective conformance checking: An approach based on integer linear programming," in *Proceedings of the 10th International Conference on Business Process Management*, 2013.

[8] ——, "Data-aware process mining: Discovering decisions in processes using alignments," in *Proceedings of the 28th ACM Symposium on Applied Computing*, 2013.

[9] W. M. P. van der Alst, "A general divide and conquer approach for process mining," in *Federated Conference on Computer Science and Information Systems*, 2013, pp. 1–10.

[10] ——, "Decomposing petri nets for process mining: A generic approach," *Distributed and Parallel Databases*, vol. 31, no. 4, pp. 471–507, 2013.

[11] W. M. P. van der Alst and B. van Dongen, "Discovering petri nets from event logs," *Transactions on Petri Nets and Other Models of Concurrency*, vol. 7480, pp. 372–422, 2013.

[12] "Process mining research tools applications," http://www.processmining.org, 2013, [Online; accessed 15-December-2013].

[13] H. Agrawal and J. Horgan, "Dynamic program slicing," in *Proceedings of the SIGPLAN Conference on Programming Language Design and Implementation.*, 1990.

[14] T. Ball and J. R. Larus, "Efficient path profiling," in *Proceedings of the IEEE/ACM Inter- national Symposium on Microarchitecture*, 1996.

[15] X. Zhang, R. Gupta, and Y. Zhang, "Precise dynamic slicing algorithms," in *Proceedings of the IEEE/ACM International Conference on Software Engineering*, 2003.

[16] X. Zhang and R. Gupta, "Matching execution histories of program version," in *Proceedings of the Joint 10th European Software Engineering Conference and 13th SIGSOFT Symposium on the Foundations of Software Engineering*, 2005.

[17] ——, "Whole execution traces and their applications," *ACM Trans. Archit. Code Optim.*, vol. 2, no. 3, pp. 301–334, 2005.

[18] E. Siever, A. Weber, S. Figgins, R. Love, and A. Robbins, *Linux in a Nutshell*. O'Reilly, 2005.

[19] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," in *Proceedings of The 1993 ACM SIGMOD Intl. Conf. on Management of Data*, 1993, pp. 207–216.

[20] H. Mannila, H. Toivonen, and A. I. Verkamo, "Discovery of frequent episodes in event sequences," *Data Mining and Knowledge Discovery*, vol. 1, pp. 259–289, 1997.

[21] M. Wojciechowski, "Discovering frequent episodes in sequences of complex events," in *Proceedings of Enlarged Fourth East-European Conference on Advances in Databases and Information Systems (ADBIS-DASFAA)*, 2000, pp. 205–214.

[22] B. Özden, S. Ramaswamy, and A. Silberschatz, "Cyclic association rules," in *Proceedings of the Fourteenth International Conference on Data Engineering*, 1998, pp. 412–421. [Online]. Available: http://dl.acm.org/citation.cfm?id=645483.656222

[23] F. Rasheeed, M. Alshalalfa, and R. Alhajj, "Efficient periodicity mining in time series databases using suffix trees," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 1, pp. 79–94, 2011.

[24] J. Han, W. Gong, and Y. Yin, "Mining segment-wise periodic patterns in time-related databases," in *Proceeding of International Conference on Knowledge Discovery and Data Mining*, 1998, pp. 214–218.

[25] K.-Y. Huang and C.-H. Chang, "SMCA: A general model for mining asynchronous periodic patterns in temporal databases."

[26] J. Yang, W. Wang, and P. S. Yu, "Mining asynchronous periodic patterns in time series data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 3, pp. 613–628, Mar. 2003. [Online]. Available: http://dx.doi.org/10.1109/TKDE.2003.1198394

[27] J.-S. Yeh, S.-C. Lin, and S.-C. Hu, "Novel algorithms for asynchronous periodic pattern mining based on 2-d linked list," *International Journal of Database Theory and Application*, vol. 5, no. 4, pp. 33–43, 2012.

[28] S. Laxman and P. S. Sastry, "A survey of temporal data mining," *Sadhana, Academy Proceedings in Engineering Sciences*, vol. 31, no. 2, pp. 173–198, 2006.

[29] J. F. Roddick and M. Spiliopoulou, "A survey of temporal knowledge discovery paradigms and methods," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, pp. 750–767, 2002.

[30] M. Zimniak, J. Getta, and W. Benn, "Mining periodic workload patterns in database audit trails," *International Journal of Database Theory and Application*, vol. 6, no. 6, pp. 63–74, 2013.

[31] D. A. Simovici and C. Djeraba, *Mathematical tools for data mining: set theory, partial orders, combinatorics*, ser. Advanced Information and Knowledge Processing. Springer, 2008.