Discovering Periodic Patterns in Irregular Time Series

by

C. Saideep, R. Uday kiran, Koji Zettsu, Philippe Fournier-Viger, Masaru Kitsuregawa, P Krishna Reddy

in

International Conference on Data Mining Workshops(ICDMW)

Report No: IIIT/TR/2019/-1

Centre for Data Engineering
International Institute of Information Technology
Hyderabad - 500 032, INDIA
November 2019

Discovering Periodic Patterns in Irregular Time Series

C. Saideep[+]
International Institute of Information Technology-Hyderabad
Telangana, India
saideep.chennupati@research.iiit.ac.in

R. Uday Kiran*
NICT, Tokyo, Japan
The University of Tokyo
Tokyo, Japan
uday_rage@tkl.iis.u-tokyo.ac.jp

Koji Zettsu
National Institute of Information and Communications Technology
Tokyo, Japan
zettsu@nict.go.jp

Philippe Fournier-Viger
Harbin Institute of Technology
Shenzhen, China
philfv8@yahoo.com

Masaru Kitsuregawa
National Institute of Informatics
The University of Tokyo
Tokyo, Japan
kitsure@tkl.iis.u-tokyo.ac.jp

P. Krishna Reddy
International Institute of Information Technology-Hyderabad
Telangana, India
pkreddy@iiit.ac.in

***Abstract*—Finding (partial) periodic patterns in time series data is a challenging problem of great importance in many applications. Due to computational reasons, most previous studies in this area have focused on the efficient discovery of periodic patterns in regular time series data. Unfortunately, these studies have limited applicability because real-world data naturally exists as an irregular time series. This paper proposes a more flexible model of periodic pattern that may be present in irregular time series. Two measures, *period* and *period-support*, were employed to determine the interestingness of a pattern in a series. The former measure captures the inter-arrival times of a pattern in a series, while the latter captures the number of periodic occurrences of a pattern in a series. A novel tree structure, called Periodic Pattern tree (PP-tree), has been introduced to record the irregular occurrences of items within the series. A pattern-growth algorithm has also been presented to find all periodic patterns from PP-tree. Experimental results demonstrate that the proposed model can find useful information, and the algorithm is efficient.**

***Index Terms*—data mining; knowledge discovery in databases, periodic patterns; pattern mining; time series data**

I. Introduction

Finding (partial) periodic patterns in time series data is a challenging problem of great importance in many applications. Due to computational reasons, most previous studies [1]–[3] in this area assumed data as regular time series and disregarded the associated timestamps. Unfortunately, this assumption is often too restrictive because the data naturally exists as an irregular time series. When confronted with this restriction in applications, researchers transformed the irregular time series into a regular time series using some form of interpolation. Such a solution is inefficient because transformation can introduce several significant and hard to quantify biases [4], primarily if there exists high irregularity in the data.

Society 5.0 is an initiative of the Government of Japan. It aims to develop a human-centered society by highly integrating physical and cyberspaces. Due to practical limitations, the data generated by several applications in Society 5.0 result in an irregular time series. Discovering periodic patterns in such irregular time series has many business applications. Few examples are as follows:

1) **(Health care analytics.)** The electronic health records (EHRs) of a patient represent an irregular time series. Figure 1 shows the hypothetical EHRs of a diabetic patient, Kotoro. A periodic pattern generated from this irregular time series data say Kotoro takes "regular insulin" at 8:00, "pre-lunch B.G.M." at 12:00 and "pre-snack B.G.M." at 22:00 hours, can be found very useful to the doctors in providing personalized recommendations. For instance, a doctor can suggest Kotoro take "pre-lunch B.G.M." around 11:00 hours instead of 12:00 hours. In the recent survey on mining EHRs [5], authors have urged for the models which can find useful information in irregular time series (or EHRs).
2) (**Transportation analytics.**) Natural disastrous, such as earth quakes, tsunamis, typhoons and floods, are very common in Japan. Monitoring congestion in a road network is an important traffic safety operation to reduce the damage loss. In this context, sensors which measure congestion are placed in many road segments. These sensors transmit information only when a state changes to conserve battery life. Thus, the data generated by these sensors often represent an irregular time series. A periodic pattern generated from such data, say congestion at 8:00 hours was observed in road_156 and road_1120, congestion at 9:00 hours was observed in road_158, and so on, can be found very useful to the users in monitoring the traffic.

Noise and irregularity are two key characteristics of big data. Noise generally refers to disturbance or deviation from the observed value, while irregularity refers to data collection

[+]Developed fast algorithm and contributed in experiments
*Proposed the model, drafted the work, and corresponding author

Day	Time	Medical activity
1	8:00	Regular insulin
	10:00	Ultralente insulin
	12:00	Pre-lunch B.G.M.
	18:00	Pre-supper B.G.M.
	22:00	Pre-snack B.G.M.
3	8:00	Regular insulin
	12:00	Pre-lunch B.G.M.
	12:00	Regular insulin
	22:00	Pre-snack B.G.M.

Day	Time	Medical activity
4	10:00	Ultralente insulin
	18:00	Pre-supper B.G.M.
	18:00	Regular insulin
	22:00	Unspecified B.G.M.
5	8:00	Regular insulin
	10:00	Ultralente insulin
	12:00	Pre-lunch B.G.M.
	18:00	Pre-supper B.G.M.
	22:00	Pre-snack B.G.M.

Fig. 1: Hypothetical EMR data constituting of various medical tests conducted on a diabetic patient. The term "B.G.M." stands for "Blood Glucose Measurement"

at uneven time stamps. Noise within the data can be handled adequately using the approaches suggested in the literature [6]. However, irregularity within the data is relatively hard to handle due to uneven gaps between the events. More important, there is an increase in the need for models that can find useful information in irregular time series [5]. With this motivation, this paper introduces a generic model of periodic pattern that may exist in an irregular time series data. Before we describe the contributions of this paper, we discuss the challenges that need to be addressed to find periodic patterns in irregular time series data.

An irregular time series allows indefinite time gaps between the consecutive events. This nature of irregular time series makes periodic pattern mining a non-trivial task. The reasons are as follows:

- Most previous studies employ measures based on $support$ to find periodic patterns. Unfortunately, such measures are inefficient to find periodic patterns in irregular time series. It is because these measures facilitate sporadic patterns to be periodic patterns.

  Example 1. In the EHRs, let "Kotoro takes regular insulin at 8:00 hours" be a pattern occurring on days 1, 3, 5, 8, 11 and 13. Let "Kotoro takes ultralente insulin at 10:00 hours" be another pattern occurring on days 1, 4, 5, 9, 10 and 15. If *minimum support* is set to 5 (days), then both patterns will be generated as periodic patterns. Clearly, the latter pattern has to be avoided because it is occurring sporadically within the data.

- Several algorithms [1], [2], [7] exist to find periodic patterns in regular time series data. These algorithms capture only the $support$ and disregard the temporal occurrence information of the items (or patterns) within the data. Henceforth, these algorithms cannot be extended to find periodic patterns in irregular time series. New algorithms that can capture the temporal occurrence information of the items within a series needs to be investigated.

This paper makes an effort to address these two challenges.

In this paper, we propose a flexible model of periodic pattern that may be present in irregular time series. Two measures, *period* and *period-support*, are employed to determine the interestingness of a pattern in a series. The former measure captures the inter-arrival time of a pattern in a series, while the latter captures the number of periodic occurrences of a pattern in a series. A novel tree structure, called Periodic Pattern tree (PP-tree), has been proposed to effectively record the irregular occurrences of items within the series. A pattern-growth algorithm, called Periodic Pattern-growth (PP-growth), has also been presented to find all periodic patterns from PP-tree. Experimental results demonstrate that the proposed model can find useful information and the algorithm is efficient.

The rest of the paper is organized as follows. Section 2 describes related work on periodic pattern mining. Section 3 describes the proposed model to find periodic patterns in irregular time series. Section 4 introduces the PP-tree and the PP-growth algorithm. Section 5 reports on experimental results. Finally, Section 6 concludes the paper with future research directions.

II. Related Work

Agrawal et al. [8] introduced a model to find frequent itemsets in a transactional database. Han et al. [2] extended the frequent itemset model to find periodic patterns in time series data. This model discovers all periodic patterns in time series data that satisfy the user-specified minimum support ($minSup$). Chen et al. [1] developed a Pattern-growth algorithm. Aref et al. [6] extended Han's model for the incremental mining of partial periodic patterns. Yang et al. [9] studied the change in periodic behavior of a pattern due to noise, and enhanced the basic model to discover a class of periodic patterns known as asynchronous periodic patterns. Yang et al. [3] used "*information gain*" as an alternative interestingness measure to $frequency$, and discovered a class of periodic patterns known as *surprising patterns*. Cao et al. [10] discussed methodologies to specify $period$ using auto-correlation. All of the above mentioned studies consider data as regular time series. On the contrary, the proposed study considers data as irregular time series and tries to find periodic patterns within the data.

Recently, the problem of finding periodic-frequent itemsets in a transactional database [11], [12] has received a great deal of attention. These studies implicitly assume that itemsets occur independently within the data. As a result, they fail to identify the sequential relationship between the periodic-frequent itemsets. For example, these studies consider "Jack reads news paper in the morning" and "Jack goes to the cafeteria in the afternoon" as two independent periodic activities (or itemsets). As a result, they fail to identify the sequential occurrence relationship between these two activities, i.e., "Jack reads news paper in the morning and goes to cafeteria in the afternoon."

Overall, the proposed model of finding periodic patterns in irregular time series is novel and distinct from the previous studies.

III. Definitions and Problem Statement

An event is a fundamental component of time series data. It represents a pair consisting of a timestamp and an item. An ordered set of events collected over a time interval (or $period$) is known as period-segment. An ordered set of period-segments is known as irregular time series.

Definition 1. Let I be the set of items (or event types). An **event** $e = (t, i)$, where $t \in \mathbb{R}^+$ is a timestamp and $i \in I$ is an item. A period-segment, PS_i, $1 \leq i$, is an ordered set of events collected within a time interval. That is. $PS_i = \{e_1, e_2, \cdots, e_n\}$, where $e_i \neq e_j$ for $1 \leq i \leq j \leq n$. A period-segment is said to be empty if there exists no event. An **irregular time series** S is an ordered collection of period-segments. That is, $S = PS_1 \cup PS_2 \cup \cdots \cup PS_m$, $1 \leq m$.

Example 2. Let $I = \{abcdefg\}$ be the set of sensors (or items) placed at different road segments. Each sensor generates a signal if congestion is observed at its respective road segment. For instance, if sensor 'a' detects congestion at timestamp 1, then it generates an event $(1, a)$. The first period-segment in Table I, i.e., $PS_1 = (1, a), (1, c), (2, b), (2, d), (2, g), (3, b), (3, d)$ represents the set of events generated in a time interval of length 3. An ordered set of all 7 period-segments in Table I represents an irregular time series.

TABLE I: Irregular time series. The term 'm' denotes the order of period-segments

m	events
1	$(1, a), (1, c), (2, b), (2, d), (2, g), (3, b), (3, d)$
2	–
3	$(1, a), (3, b), (3, e), (3, f)$
4	$(1, c), (2, e), (3, d), (3, f)$
5	$(1, a), (2, b), (2, d), (2, e), (3, b), (3, f)$
6	$(1, c), (1, g), (2, e), (3, d), (3, f)$
7	$(1, a), (1, c), (2, b), (2, d), (3, b), (3, d)$

Definition 2. Let $s = \{\widehat{e_1}, \widehat{e_2}, \cdots, \widehat{e_k}\}$, $\widehat{e_p}.i \in I$, $\widehat{e_p}.t \in (1, per)$ and $1 \leq p \leq k$, be a pattern. If a pattern s contains k individual events, then it is a k-pattern. The L-length of s represents the number of unique timestamps in s.

Example 3. Let $\gamma = \{(1, a), (3, b)\}$ be a pattern. The L-length of this pattern is 2, because it contains events with two unique timestamps.

Definition 3. (**The** $support$ **of pattern** s.) If $s \subseteq PS_j$, $1 \leq j \leq m$, it is said that s occurs in PS_j. Let PS^s_j denote the period-segment in which s has occurred in S. Let $PS^s = \{PS^s_p, \cdots, PS^s_q\}$, $1 \leq p \leq q \leq a$ be the complete set of period-segments in which s occurs. The size of PS^s represents the $support$ of s and is denoted as $SUP(s)$. That is, $SUP(s) = |PS^s|$, where $|PS^s|$ represents the number of period-segments in which s exists.

Example 4. The pattern $\gamma \subseteq PS_1$. Thus, $PS^\gamma_1 = 1$. Similarly, $PS^\gamma_2 = 3$, $PS^\gamma_3 = 5$ and $PS^\gamma_4 = 7$. The set of all period-segments in which γ exists, i.e. $PS^\gamma = \{1, 3, 5, 7\}$. The $support$ of γ, i.e., $SUP(\gamma) = |PS^\gamma| = 4$.

Definition 4. (**Frequent pattern** s) A pattern s is a **frequent pattern** if $SUP(s) \geq minSup$, where $minSup$ represents the user-specified minimum support.

Example 5. If the user-specified $minSup = 3$, then the pattern γ is a frequent pattern because $SUP(\gamma) \geq minSup$.

Definition 5. (**Interesting occurrences of** s **in** S) Let PS^s_a and PS^s_b, $p \leq a < b \leq q$, be two consecutive period-segments in PS^s. An **inter-arrival time** of s, denoted as $iat^s = (PS^s_b - PS^s_a)$. Let $IAT^s = \{iat^s_1, \cdots, iat^s_x\}$, $x = SUP(s) - 1$, be the complete set of inter-arrival times of s in various period-segments. An $iat^s_k \in IAT^s$, is considered interesting (or periodic) if $iat^s_k \leq maxIAT$, where $maxIAT$ represents the user-specified maximum inter-arrival time.

Example 6. The pattern γ consecutively appears in the period-segments 1 and 3. Therefore, an inter-arrival time of γ, i.e., $iat^\gamma_1 = 3 - 1 = 2$. Similarly, other inter-arrival times of γ are: $iat^\gamma_2 = 2$ $(= 5 - 3)$ and $iat^\gamma_3 = 2$ $(= 7 - 5)$. Thus, $IAT^\gamma = \{2, 2, 2\}$. If the user-specified $maxIAT = 2$, then iat^γ_1, iat^γ_2 and iat^γ_3 are interesting inter-arrival times of γ.

Definition 6. (**period-support of** s) Let $\widehat{IAT^s} \subseteq IAT^s$ denote the set of all interesting inter-arrival times of s in S. That is, if there exists $iat^s_k \in IAT^s : iat^s_k \leq maxIAT$, then $iat^s_k \in \widehat{IAT^s}$. The **period-support** of s, denoted as $period\text{-}support(s)$, represents the number of periodic occurrences of s in the data. That is, $period\text{-}support(s) = |\widehat{IAT^s}|$.

Example 7. Continuing with the previous example, $\widehat{SIAT^\gamma} = \{2, 2, 2\}$. Therefore, $period\text{-}support(\gamma) = |\widehat{IAT^\gamma}| = 3$.

In this paper, we have considered an *inter-arrival time* of a pattern as interesting if its value is no more than the user-specified $maxIAT$. However, our model is flexible and allows other ways to consider a segment inter-arrival time of a pattern as interesting. For instance, we can also consider a segment inter-arrival time of a pattern as interesting if its value is within $maxIAT \pm \Omega$, where $\Omega > 1$ is a constant that denotes time tolerance [13]. In this paper, we use the former definition to limit the number of input parameters and brevity.

Definition 7. (**Periodic pattern** s) The frequent pattern s is a (partial) periodic pattern if $period\text{-}support(s) \geq minPS$, where $minPS$ represents the user-defined minimum $period\text{-}support$.

Example 8. If the user-specified $minPS = 2$, then the frequent pattern γ is a periodic pattern because $period\text{-}support(\gamma) \geq minPS$.

Definition 8. (Problem Definition) Given an irregular time series (S) and the user-defined *maximum inter-arrival time* ($maxIAT$), minimum support ($minSup$) and minimum $period\text{-}support$ ($minPS$), the problem of finding periodic patterns involve finding all patterns in S that have *support* no less than $minSup$ and *period-support* no less than $minPS$. The measures, *support*, *inter-arrival time*, and $period\text{-}support$, can be expressed as percentage of m, $(m-1)$, and $(m-1)$, respectively.

The generated periodic patterns satisfy the *anti-monotonic property* (see Property 1). Thus, the proposed model practicable on very large databases. In the next section, we describe the pattern-growth algorithm to discover all periodic patterns in irregular time series.

Property 1. If $s \supset s'$, then $PS^s \subseteq PS^{s'}$. Therefore, $SUP(s) \leq SUP(s')$ and $period\text{-}support(s) \leq period\text{-}support(s')$.

IV. Periodic Pattern-Growth

Traditional pattern-growth algorithms which relay on Frequent Pattern-tree (FP-tree) [14] cannot be used for finding the periodic patterns in irregular time series. It is because FP-tree fails to record the timestamp information of the items in a series. We propose an alternative tree structure, called Periodic Pattern tree (PP-tree), to capture the timestamp information of the items within a series. Due to the structural differences of nodes in an FP-tree and PP-tree, we cannot directly apply FP-growth to mine patterns from PP-tree. Therefore, we developed another pattern-growth technique that can handle the additional features of PP-tree. We call this algorithm as Periodic Pattern-growth (PP-growth).

The PP-growth involves the following two steps: (i) compress the given irregular time series into PP-tree and (ii) discover all periodic patterns by recursively mining the PP-tree. Before we describe these two steps, we explain the structure of PP-tree.

A. The structure of PP-tree

The PP-tree is composed of a prefix-tree and a event list, called PP-list. The PP-list indicates for each distinct *event* (e), its *support* (f), *period-support* (pf), and provides a pointer to the first node in the prefix-tree representing the event (nl).

The prefix-tree in PP-tree maintains two types of nodes: **ordinary** and **tail**. An ordinary node records the temporal occurrence of an event within a period-segment. This node structure is similar to that used in an FP-tree. However, instead of maintaining the $support$ of an item, a node records the occurrence timestamp of an item in a period-segment. The structure of an ordinary node is $\langle event.item : event.timestamp \rangle$. The tail node represents an item found in the last event of any sorted period-segment. This node is designed to record the occurrence timestamp of an item in a period-segment and across period-segments. The structure of the tail node is $\langle event.item : event.timestamp : ps\text{-}list \rangle$, where **ps-list** represents the list of period-segments in which the corresponding event has occurred in a series. In other words, the tail node is an ordinary node with a ps-list. During the bottom-up mining phase of pattern-growth, the ps-list is pushed to parent nodes to determine the $support$ and $period\text{-}support$ of the patterns. Like in FP-tree, each node in a PP-tree maintains parent, children, and node traversal pointers. The key difference between an FP-tree and PP-tree is that no node in a PP-tree maintains the support count as in an FP-tree. To facilitate a high degree of compactness, events in the prefix-tree are arranged in support-descending order.

B. Constructing a PP-tree

Since periodic patterns satisfy the *anti-monotonic property*, periodic 1-patterns (or events) will play an important role in efficient discovery of periodic patterns. These periodic events can be discovered by populating the PP-list as in Algorithm 1. Briefly, the algorithm works as follows.

The scan on the first period-segment with $ps_{cur} = 1$ results in adding all its events to the PP-list as shown in Figure 2 (a) (line 9 in Algorithm 1). As no event appears in the second period-segment, the PP-list reperiod-supportns unchanged, as shown in Figure 2 (b). Figure 2 (c) shows the PP-list generated after scanning the third period-segment. Figure 2 (d) shows the PP-list generated after scanning all period-segments. Figure 2 (e) shows the sorted list of events that have $support$ and $period\text{-}support$ no less than $minSup$ and $minPS$, respectively. Let L denote this sorted list of periodic events.

After finding periodic events, we construct the prefix-tree of the PP-tree by performing another scan on the time series. Algorithms 2 and 3 describe the procedure for constructing the prefix-tree. The scan on the first period-segment generates a branch $\langle a : 1 \rangle, \langle b : 3 \rangle, \langle c : 1 \rangle, \langle d : 3 \rangle$ (format is $\langle event.item : event.timestamp \rangle$) in L order. The $tail$ node $\langle d : 3 \rangle$ carries the number of period-segments. Therefore, the tail-node of this branch is expressed as $\langle d : 3 : 1 \rangle$ (format is $\langle event.item : event.timestamp : period\text{-}segments \rangle$). The PP-tree generated after scanning the first period-segment is shown in Figure 3(a). The scan on the second period-segment will not result in any branch as there exists no event in this segment. The scan on the third period-segment generates a branch $\langle a : 1 \rangle, \langle b : 3 \rangle, \langle f : 3 : 3 \rangle$. However, this branch shares the common prefix-path $\langle a : 1 \rangle, \langle b : 3 \rangle$ with the already existing branch. Therefore, we simply create a node $\langle f : 3 : 3 \rangle$, and connect it as a child node of $\langle b : 3 \rangle$, as shown in Figure 3(b). A similar process is repeated for the reperiod-supportning period-segments and the PP-tree is generated accordingly. To facilitate tree traversal, node-links are period-supportntained in the PP-tree as in an FP-tree. Figure 3(c) shows the complete PP-tree generated after scanning all period-segments. **It can be observed that PP-tree achieves memory efficiency by recording period-segment information only at the tail-nodes rather than at all node.**

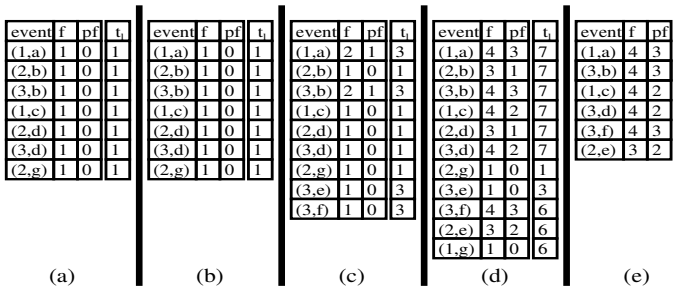


Fig. 2: Construction of PP-list. (a) After scanning first period-segment, (b) after scanning second period-segment, (c) after scanning third period-segment, (d) after scanning all time series data, and (e) sorted list of periodic events after removing both aperiodic and infrequent events

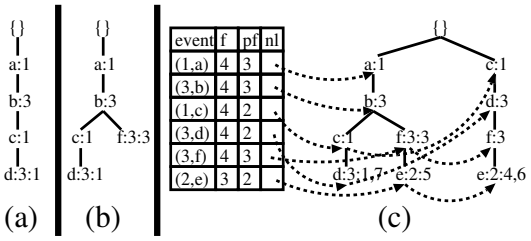


Fig. 3: Construction of PP-tree. (a) After scanning first period-segment, (b) after scanning third period-segment, (c) final PP-tree generated after scanning all period-segments

C. Mining a PP-tree

Algorithm 4 describes the procedure for finding periodic patterns from PP-tree. As the calculation of *period-support* is a simple procedure, We are not providing it in this paper.

Starting from a length-1 pattern (as an initial suffix pattern), construct its conditional pattern base (a sub-database, which consists of the set of prefix paths in a PP-tree with the suffix pattern), construct its conditional PP-tree, and perform mining recursively on the tree. Achieve pattern-growth by concatenating the suffix pattern with the periodic patterns generated from a conditional PP-tree. Next, the suffix pattern has to be completely removed from the PP-tree by pushing its **ps-list** to the respective parent nodes. Please note that this step is novel as it does not exist in current FP-growth-like algorithms. All the above steps are repeated until the PP-list is empty.

We start with the bottom-most item in the PP-list. Since $(2, e)$ is the bottom-most item, each node with $event.item = e$ and $event.timestamp = 2$ (i.e., $\langle e : 2\rangle$) in the PP-tree must be a tail-node. While constructing the conditional pattern base of $\langle e : 2\rangle$, we map the ps-list of every node $\langle e : 2\rangle$ to all items in the respective path explicitly in a temporary array (one for each event). This facilitates the calculation of $support$ and $period\text{-}support$ for each event in the conditional pattern base of $\langle e : 2\rangle$. Figure 4(a) shows the conditional pattern base of $\langle e : 2\rangle$. The conditional PP-tree of $\langle e : 2\rangle$ is constructed with only events having a $support$ no less than $minSup$ and a *period-support* no less than $minPS$. Figure 4(b) shows the conditional PP-tree for $\langle e : 2\rangle$. The concatenation of $\langle e : 2\rangle$ with $\langle f : 3 : 4, 5, 6\rangle$ will result in a periodic pattern $\{(2, e), (3, f)\}$ with $support = 3$ and *period-support = 2*. To enable the construction of the prefix-tree for the next event in the PP-list, the event $(e : 2)$ is completely pruned from the original PP-tree by pushing its ps-lists to the respective parent nodes (see Figure 4(c)). The entire process is repeated until the PP-list in the PP-tree is empty. The correctness of our algorithm is based on Property 2 and shown in Lemma 1.

An advantage of PP-growth algorithm is that its distributed version can be easily developed using map-reduce framework. Due to page limitation, we are not discussing this algorithm in this paper. For more information, please refer to distributed

Algorithm 1 PP-List (S: time series data, I: set of items, $maxIAT$: maximum inter-arrival time, $minSup$: minimum support and $minPS$: minimum period-support)

1: Let t_l be a temporary array that records the *timestamp* of the last appearance of each event in S. Let ps_{cur} denote the number (or position) of current period-segments PS_j.
2: **for** each period-segment $PS_j \in S$ **do**
3: **for** each event $e \in PS_j$ **do**
4: **if** e exists in PP-list **then**
5: **if** $(ps_{cur} - t_l^e) \leq maxIAT$ **then**
6: Set ++pf^e.
7: **end if**
8: Set ++f^e and $t_l^e = ps_{cur}$.
9: **else**
10: Add e to the PP-list with $f^e = 1$, $pf^e = 0$, and $t_l^e = ps_{cur}$.
11: **end if**
12: **end for**
13: **end for**
14: **for** each event $e \in PP\text{-}list$ **do**
15: **if** $f^e < minSup$ or $pf^e < minPS$ **then**
16: Remove e from the PP-list.
17: **end if**
18: **end for**
19: Sort the reperiod-supportning events of the PP-list in descending order of their frequencies.

Algorithm 2 PP-Tree (S, PP-list)

1: Create the root node in the PP-tree, $Tree$, and label it "$null$".
2: **for** $j = 1$: $j \leq m$; $++j$ **do**
3: Select the periodic events in PS_j and sort them in L order. Let the sorted periodic event list be $[e|E]$, where e is the first event and E is the reperiod-supportning list. Call $insert\_tree([e|E], j, Tree)$.
4: **end for**
5: call PP-growth ($Tree$, $null$);

Algorithm 3 insert_tree($[e|E]$, ps_{cur}, $Tree$)

1: **while** E is non-empty **do**
2: **if** $Tree$ has a child N such that $e.item \neq N.item$ and $e.timestamp \neq N.timestamp$ **then**
3: Create a new node N. Let its parent-node be linked to $Tree$. Let its node-link be linked to only those nodes that have the same item and timestamp via the node-link structure. Remove e from E.
4: **end if**
5: **end while**
6: Add ps_{cur} to the leaf node.

FP-growth algorithms.

Property 2. A tail-node in a PP-tree maintains the occurrence information for all the nodes in the path (from that tail-node to the root), at least in the transactions in its ps-list.

Lemma 8. Let $B = \{b_1, b_2, \cdots, b_n\}$ be a branch in a PP-tree where b_n is a tail-node carrying the ps-list of the branch. If the ps-list is pushed-up to node b_{n-1}, then b_{n-1} maintains the occurrence information of the path $B' = \{b_1, b_2, \cdots, b_{n-1}\}$ for the same set of transactions in the ps-list without any loss.

Proof. According to Property 2, b_n maintains the occurrence information of path B' at least in the period-segments in its ps-list. Therefore, the same ps-list at node b_{n-1} maintains the same transaction information for B' without any loss.

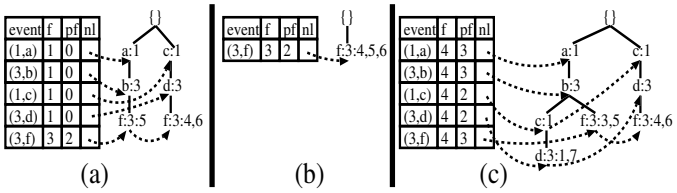


Fig. 4: Mining PP-tree. (a) Conditional pattern base of $\langle e : 2\rangle$, (b) conditional PP-tree of $\langle e : 2\rangle$, and (c) PP-tree generated after pruning the event $(2, e)$

V. Experimental evaluation

Since there exists no algorithm to find periodic patterns in an irregular time series, we only evaluate our model using synthetic and real-world databases. We show that the $minPS$ constraint facilitates us to prune many aperiodic frequent patterns from the data. We also show that PP-growth is not only memory and runtime efficient, but also scalable as well. We also demonstrate the usefulness of the proposed model with two real-world applications.

A. Experimental setup

The PP-growth algorithm was implemented in Python3 and run with Ubuntu 18.0 on a 2.66 GHz Intel i5 machine with 8 GB of RAM. Both synthetic (T10I4D100K and T10I4D1000K) and real-world (Road and Patient-1) databases have been used for the experiments.

Algorithm 4 PP-growth ($Tree$, α)

1: **for** each event e_i in the header of $Tree$ **do**
2: Generate pattern $\beta = e_i \cup \alpha$. Traverse $Tree$ using the node-links of β, and construct an array, PS^{β}, which represents the list of period-segments in which β has appeared in S. Construct β's conditional pattern base and β's conditional PP-tree $Tree_{\beta}$ if $PS^{\beta}.length \geq minSup$ and $period\text{-}support(PS^{\beta}) \geq minPS$.
3: **if** $Tree_{\beta} \neq \emptyset$ **then**
4: call PP-growth ($Tree_{\beta}$, β);
5: **end if**
6: Remove e_i from the $Tree$ and push the e_i's ps-list to its parent nodes.
7: **end for**

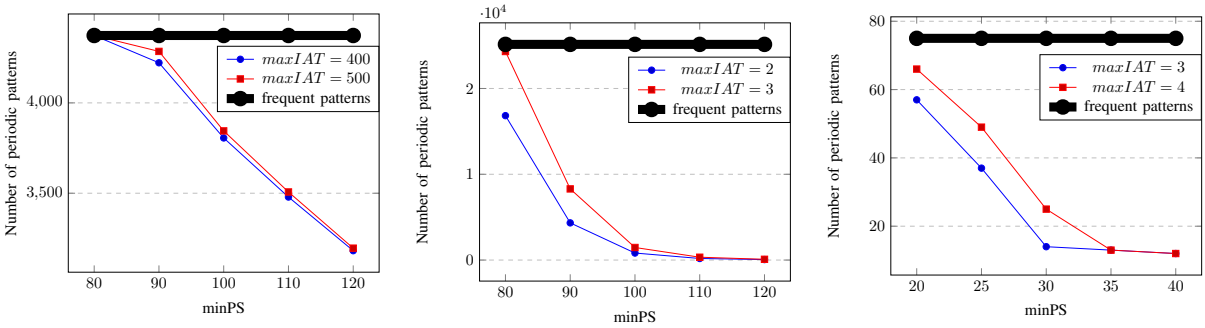
The synthetic transactional databases, **T10I4D100K** and **T10I4D1000K**, have been generated using the IBM Almaden Quest dataset generator [8]. These two databases have been widely used for evaluating pattern mining algorithms. The **T10I4D100K** database contains 963 items and 100,000 transactions. The minimum, average, and maximum length of a transaction in **T10I4D100K** database are 1, 11.01 and 31, respectively. The **T10I4D1000K** database contains 30,388 items and 983,155 transactions. The minimum, average, and maximum length of a transaction in **T10I4D1000K** database are 2, 11.1 and 32, respectively. Both transactional databases have been converted into time series databases by considering a sequence as a set of consecutive 10 transactions (i.e., $period = 10$).

Electronic Health Records (EHRs) of diabetic patients have been obtained from the UCI repository [15]. In this paper, we confine our experiment to EHRs of a patient whose identifier is 1. We call this database as "Patient-1." A sequence represents a set of medical tests underwent of a patient in a single day. The **Patient-1** was collected from 21-04-1991 to 03-09-1991 (i.e., 136 days). This database contains 77 items and 135 transactions. The minimum, average, and maximum length of a transaction in **Patience-1** database are 3,6.95, 13, respectively.

The **Road** database is provided by Japan Road Traffic Information Center (JRTIC). Each transaction in this database contains a set of road segments where observed congestion is more than 300 meters in an hour. Each sequence contained a set of transactions observed in a hour. The **Road** database contains 2231 items and 4373 transactions. The minimum, average, and maximum length of a transaction in this database are 10, 50.7, 228, respectively.

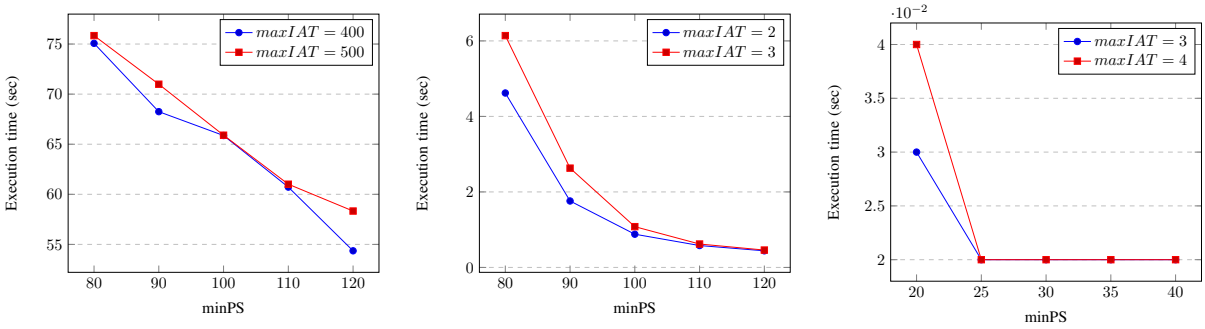
B. Generation of periodic patterns

Fig. 5a, 5b and 5c show the number of periodic patterns generated in T10I4D100K, Road and Patient databases, respectively. The thick black line in these figures represent the number frequent patterns generated from the respective databases when $minPS$ is set to zero. The $minSup$ count



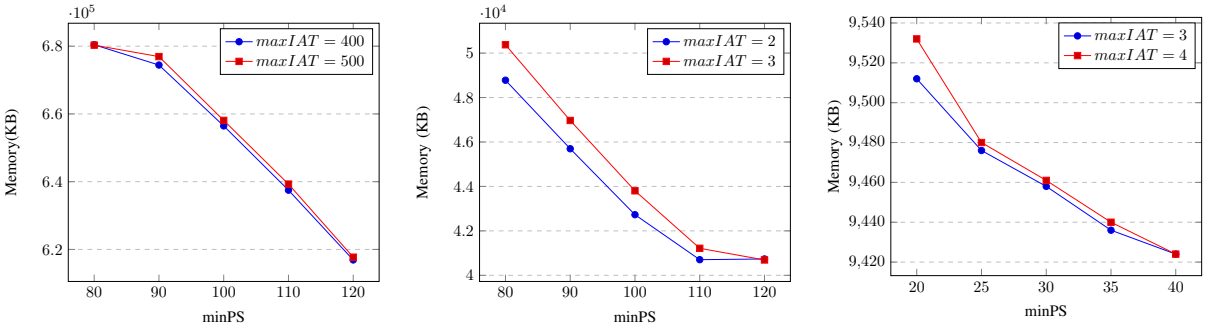
(a) T10I4D100K (b) Road (c) Patient-1

Fig. 5: Number of frequent patterns and periodic patterns generated in various databases at different $minPS$ and $maxIAT$ values



(a) T10I4D100K (b) Road (c) Patient-1

Fig. 6: Runtime requirements of PP-growth in various databases



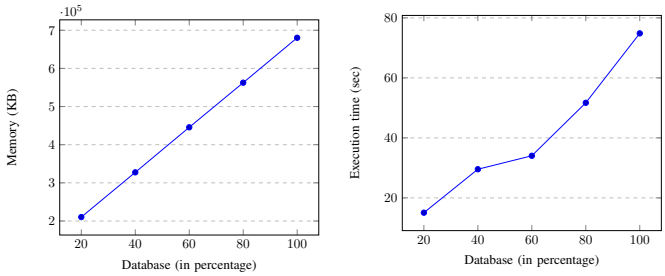
(a) T10I4D100K (b) Road (c) Patient-1

Fig. 7: Memory consumed by PP-growth

in T10I4D100k, Road and Patient-1 databases has been set at 90, 90, and 25, respectively. Please note that all further experimental results have been reported at the above $minSup$ values. The following two observations can be drawn from these figures: (i) The constraint, $minPS$, has negative effect on the number of periodic patterns being generated. It is because many patterns fail to satisfy the increased $minPS$ value. (ii) The constraint, $maxIAT$, has positive effect on the number of periodic patterns being generated. It is because the increase in $maxIAT$ typically increases the periodic-support of a pattern, thereby, facilitating more patterns to be periodic patterns. (iii) It can be observed that the usage of $minPS$ and $maxIAT$ constraints have facilitated us to prune many aperiodic frequent patterns from the data.

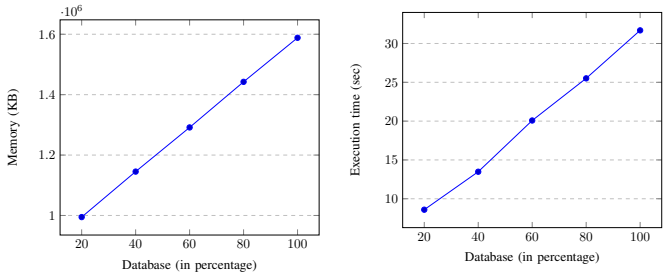
Fig. 6a, 6b and 6c show the runtime requirements of PP-growth on T10I4D100K, Road, and Patient databases, respectively. The runtime involves both constructing and mining a PP-tree. The changes in the $maxIAT$ and $minPS$ values shows a similar effect on runtime consumption as in the generation of periodic patterns. More importantly, PP-growth discovered the complete set of periodic patterns in a reasonable time even for low $minPS$ and high $maxIAT$ values.

Fig. 7a 7b and 7c show the memory requirements of PP-growth on T10I4D100K, Road, and Patient databases, respec-



(a) Memory (b) Runtime

Fig. 8: Scalability of PP-growth on T10I4D100K database



(a) Memory (b) Runtime

Fig. 9: Scalability of PP-growth on T10I4D1000K database

tively. It can be observed from these figures that even at high $maxIAT$ and low $minPS$ values, the PP-tree consumes a small amount of memory since it maintains information pertaining to period-segments only in tail-nodes.

C. Scalability test

We study the scalability of PP-growth on execution time and required memory by varying the length of time series data. We use T10I4D1000K and T25I6D1000K data sets for this experiment. We divided each data set into five portions of 0.2 million transactions in each part. Then we investigated the performance of PP-growth after accumulating each portion with previous parts while performing periodic pattern mining each time. Each period-segment contained 10 transactions. The $maxIAT$, $minSup$ and $minPS$ values have been set at 5000 (in count), 0.25% and 0.25%, respectively.

Fig. 8a and 8b respectively show the memory and runtime requirements of PP-growth algorithm in T10I4D100K database. Fig. 9a and 9b respectively show the memory and runtime requirements of PP-growth algorithm in T19I4D1000K database. It is clear from the graphs that as the database size increases, overall tree construction and mining time, and memory requirement increase. However, PP-tree shows stable performance of about linear increase of runtime and memory consumption with respect to the data size. Therefore, it can be observed from the scalability test that PP-growth can mine periodic patterns over large data sets and distinct items with considerable amount of runtime and memory.

D. A case study on health-care data

Table II lists interesting patterns discovered in Patient-1 database. These patterns represent the regular medical practices followed by a patient. The following interesting information can be derived from these three patterns: (i) If the Patient-1 takes 'NPH insulin dose' in the morning, then he takes 'Regular insulin dose' only once in the entire day. (ii) If the Patient-1 takes 'Pre-breakfast blood glucose measurement' and 'NPH insulin dose' at 8:00 hrs (instead of at 7:00 hrs), then he takes 'Regular insulin dose' at 12:00 hrs. Such information is useful for doctors to provide personalized recommendations to patients.

E. A case study on Road database

Table III lists some of the interesting patterns generated from the Road database. The following observations can be drawn from these figures: (i) In the morning, congestion can be regularly observed on the road whose identifier is 1683. (ii) During afternoon, congestion is often observed on the roads whose identifiers are 1502 and 517. (iii) In the evening, congestion is often observed on the road whose identifier is 1181. (iv) In the night, congestion is often observed on the road whose identifier is 1473. This information regarding the regular congestion of roads at different time intervals of a day can be found to be very useful for the users in urban planning and monitoring traffic at the time of disasters.

VI. Conclusions and Future Work

We have presented a generic model of periodic pattern that may exist in an irregular time series data. The model aims to find all patterns that have exhibited partial cyclic repetitions within the data. The generated patterns satisfy the anti-monotonic property. This property facilitates our model practicable in real-world applications. A novel pattern-growth algorithm has been presented to find all periodic patterns in a series. Experimental results have shown that the proposed model can find useful information, and that the algorithm is efficient.

This paper has studied the discovery of periodic patterns in irregular time series. The proposed model can be extended to incremental mining of irregular time series and data streams. As there exists no universally acceptable best measure to judge the interestingness of patterns in any data, it is interesting to investigate alternative measures of $period\text{-}support$.

References

[1] S.-S. Chen, T. C.-K. Huang, and Z.-M. Lin, "New and efficient knowledge discovery of partial periodic patterns with multiple minimum supports," *J. Syst. Softw.*, vol. 84, pp. 1638–1651, Oct. 2011.

[2] J. Han, G. Dong, and Y. Yin, "Efficient mining of partial periodic patterns in time series database," in *ICDE*, pp. 106–115, 1999.

[3] R. Yang, W. Wang, and P. Yu, "Infominer+: mining partial periodic patterns with gap penalties," in *ICDM*, pp. 725–728, 2002.

[4] A. Eckner, "A framework for the analysis of unevenly-spaced time series data," 2011. http://www.eckner.com/papers/unevenly_spaced_time_series_analysis.pdf.

[5] P. Yadav, M. Steinbach, V. Kumar, and G. Simon, "Mining electronic health records (ehrs): A survey," *ACM Computing Surveys*, vol. 50, 1 2018.

TABLE II: Some interesting patterns discovered in Patient-1 database.

S.No.	Pattern	*support*	*PS*
1	{(NPH insulin dose,7:00hrs),(Regular insulin dose,7:00hrs), (Pre-breakfast blood glucose measurement,7:00hrs) (Pre-supper blood glucose measurement,17:00hrs),(Post-supper blood glucose measurement,17:00hrs)}	34	23
2	{(Regular insulin dose,7:00hrs), (Pre-breakfast blood glucose measurement,7:00hrs), (Regular insulin dose,17:00hrs) (Pre-supper blood glucose measurement, 17:00hrs)}	34	23
3	{Pre-breakfast blood glucose measurement,8:00hrs),(NPH insulin dose,8:00hrs),(Regular insulin dose,12:00hrs)}	25	20

TABLE III: Some interesting patterns discovered in Road database.

S.No.	Pattern	*support*	*period-support*
1	{(1683, 9:00 hrs), (1683, 10:00 AM), (1181, 16:00 hours), (1181, 17:00PM)},	92	85
2	{(1502,12:00 hrs), (517, 11:00 hrs), (517, 12:00 hrs), (1181, 16:00 hrs), (1181, 17:00 hrs)}	103	102
3	{(1502, 11:00 hrs), (1502, 12:00 hrs), (517, 12:00 hrs), (1473, 20:00 hrs),(1473, 22:00 hrs)}	90	87

[6] W. G. Aref, M. G. Elfeky, and A. K. Elmagarmid, "Incremental, online, and merge mining of partial periodic patterns in time-series databases," *IEEE TKDE*, vol. 16, pp. 332–342, Mar. 2004.

[7] J. Han, W. Gong, and Y. Yin, "Mining segment-wise periodic patterns in time-related databases.," in *KDD*, pp. 214–218, 1998.

[8] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," in *SIGMOD*, pp. 207–216, 1993.

[9] J. Yang, W. Wang, and P. S. Yu, "Mining asynchronous periodic patterns in time series data.," *IEEE Trans. Knowl. Data Eng.*, pp. 613–628, 2003.

[10] H. Cao, D. Cheung, and N. Mamoulis, "Discovering partial periodic patterns in discrete data sequences," in *Advances in Knowledge Discovery and Data Mining*, vol. 3056, pp. 653–658, 2004.

[11] S. K. Tanbeer, C. F. Ahmed, B. S. Jeong, and Y. K. Lee, "Discovering periodic-frequent patterns in transactional databases," in *PAKDD*, pp. 242–253, 2009.

[12] R. U. Kiran, J. N. Venkatesh, P. Fournier-Viger, M. Toyoda, P. K. Reddy, and M. Kitsuregawa, "Discovering periodic patterns in non-uniform temporal databases," in *PAKDD II*, pp. 604–617, 2017.

[13] S. Ma and J. Hellerstein, "Mining partially periodic event patterns with unknown periods," in *ICDE*, pp. 205–214, 2001.

[14] J. Han, H. Cheng, D. Xin, and X. Yan, "Frequent pattern mining: Current status and future directions," *DMKD*, vol. 14, no. 1, 2007.

[15] M. Kahn, *Diabetes Data Set*, 1994. Available at http://archive.ics.uci.edu/ml/datasets/Diabetes.