Discovering Partial Periodic Itemsets in Temporal Databases

R. Uday Kiran*
National Institute of Information and Communications Technology
Tokyo, Japan
uday_rage@tkl.iis.u-tokyo.ac.jp

Haichuan Shang†
National Institute of Information and Communications Technology
Tokyo, Japan
shang@tkl.iis.u-tokyo.ac.jp

Masashi Toyoda
University of Tokyo
Tokyo, Japan
toyoda@tkl.iis.u-tokyo.ac.jp

Masaru Kitsuregawa‡
University of Tokyo
Tokyo, Japan
kitsure@tkl.iis.u-tokyo.ac.jp

ABSTRACT

A temporal database is a collection of transactions, ordered by their timestamps. Discovering partial periodic itemsets in temporal databases has numerous applications. However, to the best of our knowledge, no work has considered finding these itemsets in temporal databases, despite that this type of data is very common in real-life. Discovering partial periodic itemsets in temporal databases is challenging. It requires defining (*i*) an appropriate measure to assess the periodic interestingness of itemsets, and (*ii*) an algorithm to efficiently find all partial periodic itemsets. While a pattern-growth algorithm can be employed for the second sub-task, the first sub-task has not been addressed. Moreover, how these two tasks are combined has significant implications. In this paper, we address this challenge. We introduce a model to find partial periodic itemsets in temporal databases. A new measure, called *periodic-frequency*, has been proposed to determine the periodic interestingness of itemsets by taking into account their number of cyclic repetitions in the entire data. Moreover, the paper introduces a pattern-growth algorithm to discover all partial periodic itemsets. Experimental results demonstrate that our model is efficient.

CCS CONCEPTS

• **Information systems → Association rules**;

KEYWORDS

data mining, periodic patterns, pattern mining

*This author is also affiliated to the University of Tokyo, Tokyo, Japan.
†This author is also affiliated to the University of Tokyo, Tokyo, Japan.
‡This author is also affiliated to the National Institute of Informatics, Tokyo, Japan.

SSDBM '17, June 27-29, 2017, Chicago, IL, USA

ACM ISBN 978-1-4503-5282-6/17/06…$15.00
https://doi.org/http://dx.doi.org/10.1145/3085504.3085535

ACM Reference format:
R. Uday Kiran, Haichuan Shang, Masashi Toyoda, and Masaru Kitsuregawa. 2017. Discovering Partial Periodic Itemsets in Temporal Databases. In *Proceedings of SSDBM '17, Chicago, IL, USA, June 27-29, 2017,* 6 pages.
https://doi.org/http://dx.doi.org/10.1145/3085504.3085535

1 INTRODUCTION

A temporal database is a collection of transactions and their timestamps. Three key properties of a temporal database are: (*i*) all transactions are ordered by their timestamps (*ii*) a time gap can exist in-between consecutive transactions and (*iii*) multiple transactions can share a common timestamp. These three properties differentiate a temporal database from a widely studied transactional database, which basically represents a collection of transactions. Temporal data is naturally produced in industrial and scientific domains. Examples include market-basket data, Twitter data, accidents data and sensor networks data.

Partial periodic itemsets[1] are an important class of regularities that exist in a temporal database. A partial periodic itemset is something persistent and predictable that appears in the data. Finding partial periodic itemsets is thus useful to understand data. For example, it was revealed in our present study on Twitter data related to the Great East Japan Earthquake (GEJE)[2] that approximately 44% of the event keywords found by a supervised event detection algorithm [12] can also be discovered as partial periodic itemsets. The proposed study thus may be used as an unsupervised learning technique to generate some prior knowledge about event keywords and their associations in Twitter data.

The task of finding partial periodic itemsets has two important sub-tasks: (*i*) determining the periodic interestingness of itemsets, and (*ii*) finding all partial periodic itemsets in a given database. While a variation of pattern-growth algorithms could be employed for the second sub-task, the first sub-task is non-trivial because of the following reasons:

(1) Current partial periodic pattern[3] models [3, 4, 17–19] do not take into account the information about the temporal occurrences of items in a dataset.

[1]A set of items represent an itemset.
[2]GEJE happened on 11^{th} March 2011
[3]A set of itemsets represent a pattern.

(2) Since a temporal database allows time gaps between consecutive transactions and transactions to share a common timestamp, the periodic interestingness of an itemset has to be determined by taking into account not only its $frequency$, but also its *inter-arrival times* in a database. Unfortunately, current measures assess the interestingness of an itemset by only taking its $frequency$ into account [14]. We need to investigate new measure(s) to assess the interestingness of itemsets by taking into account both their $frequency$ and *inter-arrival times* in the data.

Moreover, how to combine the two aforementioned tasks has significant implications.

This paper addresses this challenge. It presents a model to discover partial periodic itemsets in temporal databases. A new measure, called *periodic-frequency*, is proposed to determine the periodic interestingness of an itemset in the data. Unlike the existing $frequency$-based measures, the proposed measure assess the interestingness of an itemset by taking its number (or proportion) of cyclic repetitions into account. An inter-arrival time of an itemset is considered cyclic (or periodic) if it is no more than a user-specified *period*. We also propose a pattern-growth algorithm, called Partial Periodic Pattern-growth (3P-growth), to discover the complete set of partial periodic itemsets. Experimental results demonstrate that the proposed algorithm is efficient.

The rest of paper is organized as follows. Section 2 describes the related work. Section 3 describes the proposed model of partial periodic itemsets. Section 4 introduces our algorithm to find all partial periodic itemsets in a temporal database. Section 5 reports on experimental results. Finally, Section 6 concludes the paper with future research directions.

2 REVIEW OF LITERATURE

Agrawal et al. [1] introduced a model to find frequent itemsets in a transactional database. Han et al. [4] enhanced the frequent itemset model to discover partial periodic patterns in time series data. Aref et al. [3] extended the Han's model to incremental mining of partial periodic patterns. Yang et al. [18] used *information gain* as an alternative interestingness measure for $frequency$ to find partial periodic patterns. Yang et al. [17] studied the change in periodic behavior of a pattern due to the influence of noise, and discussed a model to find a class of partial periodic patterns known as asynchronous periodic patterns. Zhang et al. [19] enhanced the basic model to discover periodic patterns in character sequences like protein data. All of the above mentioned studies consider time series as a symbolic sequence (or events occurring at a fixed time interval). As a result, these studies does not take into account the actual temporal information of events within a sequence [10]. On the contrary, our study finds partial periodic itemsets by taking into account the temporal occurrence information of the transactions in the data. More importantly, since transactions in a temporal database can share a common timestamp, these databases cannot be represented as a time series.

Tanbeer et al. [15] described a model to find full periodic-frequent itemsets in a transactional database. This model finds all frequent itemsets that have exhibited complete (or full) cyclic repetitions in the data. Amphawan et al. [2] investigated the problem of finding top-k full periodic-frequent itemsets in a transactional database. We have also studied the problem of finding full periodic-frequent itemsets in a transactional database [6–9, 13, 16]. This problem of finding full periodic-frequent itemsets greatly simplifies the design of the model because there is no need of any measure to determine the partial periodic interestingness of an itemset. More importantly, these studies also consider transactional database as a symbolic sequence of transactions (occurring at fixed time interval) and ignore the temporal occurrence information of the transactions in a database. Our study finds (partial) periodic itemsets by taking into account the temporal occurrence information of the transactions in a database.

Recently, we have discussed a model to find partial periodic-frequent itemsets in a transactional database. The partial periodic-frequent itemsets does not satisfy the anti-monotonic property [1]. As a result, the model is computationally expensive to use in real-world very large databases. Moreover, this model cannot handle temporal databases as multiple transactions can share a common timestamp.

3 PROBLEM DEFINITION

Let $I = \{i_1, i_2, \cdots, i_n\}$ be the set of 'n' items appearing in a database. A set of items $X \subseteq I$ is called an **itemset**. An itemset containing k items is called a k-itemset. The length of this itemset is k. A transaction tr consists of transaction identifier, timestamp and an itemset. That is, $tr = (tid, ts, Y)$, where tid represents the transactional identifier, $ts \in \mathbb{R}$ represents the transaction time (or timestamp) and Y is an itemset. A temporal database TDB is an ordered collection of transactions, i.e. $TDB = \{tr_1, tr_2, \cdots, tr_m\}$, where $m = |TDB|$ represents the database size (or the total number of transactions). Let ts_{min} and ts_{max} denote the minimum and maximum timestamps of all transactions in TDB, respectively. For a transaction $tr = (tid, ts, Y)$, such that $X \subseteq Y$, it is said that X occurs in tr and such a timestamp is denoted as ts^X. Let $TS^X = (ts^X_a, ts^X_b, \cdots, ts^X_c)$, $ts_{min} \leq ts^X_a \leq ts^X_b \leq ts^X_c \leq ts_{max}$, be the **ordered list of timestamps** of transactions in which X appears in TDB. The number of transactions containing X in TDB (i.e., the size of TS^X) is defined as the $frequency$ of X and denoted as $freq(X)$. That is, $freq(X) = |TS^X|$.

Example 3.1. Table 1 shows a temporal database with $I = \{a, b, c, d, e, f, g\}$. The set of items '$a$' and '$b$,' i.e., $\{a, b\}$ (or 'ab') is an itemset. This itemset contains 2 items. Therefore, it is a 2-itemset. The length of this itemset is 2. In the first transaction, $tr_1 = (101, 1, abg)$, '101' represents the tid of the transaction, '1' represents the timestamp of this transaction and 'abg' represents the items occurring in this transaction. Other transactions in this database follow the same representation. This database contains 14 transactions. Therefore, $m = 14$. The minimum and maximum timestamps in this database are 1 and 12, respectively. Therefore, $ts_{min} = 1$ and $ts_{max} = 12$. The itemset 'ab' appears in the transactions whose timestamps are 1, 3, 5, 9, 11, 12 and 12. Therefore, $TS^{ab} = \{1, 3, 5, 9, 11, 12, 12\}$. The $frequency$ of 'ab,' i.e., $freq(ab) = |TS^{ab}| = 7$.

Definition 3.2. **(Periodic appearance of itemset X.)** Let ts^X_j, $ts^X_k \in TS^X$, $ts_{min} \leq ts^X_j \leq ts^X_k \leq ts_{max}$, denote any two consecutive timestamps in TS^X. The time difference between ts^X_k and ts^X_j is

Table 1: Temporal database

tid	*ts*	*items*
101	1	*abg*
102	1	*acd*
103	3	*ab*
104	4	*aef*
105	5	*abg*
106	6	*cd*
107	7	*bg*
108	8	*cdef*
109	9	*abef*
110	9	*ade*
111	10	*cdg*
112	11	*abef*
113	12	*abcd*
114	12	*abcd*

referred as an **inter-arrival time** of X, and denoted as $iat_p^X, p \geq 1$. That is, $iat_p^X = ts_k^X - ts_j^X$. Let $IAT^X = \{iat_1^X, iat_2^X, \cdots, iat_{freq(X)-1}^X\}$, be the list of all inter-arrival times of X in TDB. An inter-arrival time of X is said to be **periodic** (or interesting) if it is no more than the user-specified *period* (*per*). That is, an $iat_i^X \in IAT^X$ is said to be **periodic** if $iat_i^X \leq per$.

Example 3.3. The itemset '*ab*' has initially appeared at the timestamps of 1 and 3. The time gap in-between these two timestamps gives an inter-arrival time of '*ab*.' That is, $iat_1^{ab} = 2\ (= 3 - 1)$. Similarly, other inter-arrival times of '*ab*' are $iat_2^{ab} = 2\ (= 5 - 3)$, $iat_3^{ab} = 4\ (= 9 - 5)$, $iat_4^{ab} = 2\ (= 11 - 9)$, $iat_5^{ab} = 1\ (= 12 - 11)$ and $iat_6^{ab} = 0\ (= 12 - 12)$. Therefore, the resultant $IAT^{ab} = \{2, 2, 4, 2, 1, 0\}$. If the user-specified $per = 2$, then $iat_1^{ab}, iat_2^{ab}, iat_4^{ab}$, iat_5^{ab} and iat_6^{ab} are considered as the periodic occurrences of '*ab*' in the data. The iat_3^{ab} is considered as an aperiodic occurrence of '*ab*' because $iat_3^{ab} \not\leq per$.

Definition 3.4. **(Periodic-frequency of itemset X.)** Let $\widehat{IAT^X}$ be the set of all inter-arrival times in IAT^X that are no more than *per*. That is, $\widehat{IAT^X} \subseteq IAT^X$ such that if $\exists iat_k^X \in IAT^X : iat_k^X \leq per$, then $iat_k^X \in \widehat{IAT^X}$. The *periodic-frequency* of X, denoted as $PF(X) = |\widehat{IAT^X}|$.

Example 3.5. Continuing with the previous example, $\widehat{IAT^{ab}} = \{2, 2, 2, 1, 0\}$. Therefore, the *periodic-frequency* of '*ab*,' i.e. $PF(ab) = |\widehat{ab}| = |\{2, 2, 2, 1, 0\}| = 5$.

The *periodic-frequency*, as defined above, determines the number of periodic occurrences of an itemset in the database. It can be observed that this measure determines the interestingness of an itemset by taking into account both its $frequency$ and *inter-arrival times* in the data. The *inter-arrival times* of an itemset and the *period* can be expressed in percentage of $(ts_{max} - ts_{min})$. The *periodic-frequency* of an itemset can also be expressed in percentage of $|TDB| - 1$, where $|TDB| - 1$ represent the maximum *periodic-frequency* an itemset can have in the database. In this paper, we use the above definitions of *inter-arrival times*, *period* and *periodic-frequency* for brevity.

Definition 3.6. (Partial periodic itemset X.) An itemset X is a partial periodic itemset if $PF(X) \geq minPF$, where *minPF* represents the user-specified minimum periodic-frequency.

Example 3.7. Continuing with the previous example, if the user-specified $minPF = 2$, then '*ab*' is a partial periodic itemset because $PF(ab) \geq minPF$.

(Problem definition.) Given a temporal database (TDB), a set of items (I), *period* (*per*) and *minimum periodic-frequency* (*minPF*), the problem of finding partial periodic itemsets involve discovering all itemsets in TDB that have *periodic-frequency* no less than *minPF*.

The partial periodic itemsets discovered by the proposed model satisfy the *downward closure property* [1]. The correctness of our statement is straight forward to prove from Property 1.

Property 1. *If $X \subseteq Y$, then $TS^X \supseteq TS^Y$. Therefore, $PF(X) \geq PF(Y)$.*

4 3P-GROWTH

The proposed 3P-growth algorithm involves the following two steps: (*i*) compress the database into partial periodic pattern tree (3P-tree) and (*ii*) recursively mine the 3P-tree to find all partial periodic itemsets.

4.1 Construction of 3P-tree

A 3P-tree has two components: a 3P-list and a prefix-tree. The 3P-list consists of each distinct *item* (*i*), *periodic-frequency* (pf) and a pointer pointing to the first node in the prefix-tree carrying the item. The structure of prefix-tree is same as the prefix-tree in Periodic-Frequent tree (PF-tree) [15].

Since partial periodic itemsets satisfy the anti-monotonic property, partial periodic items (or 1-itemsets) will play an important role in effective mining of these itemsets. Algorithm 1 describes the steps for finding partial periodic items in a temporal database. Figure 1(a)-(c) show the construction of 3P-list after scanning the first, second and every transaction in the database, respectively. Figure 1(d) shows the 3P-list containing partial periodic items in descending order of their pf value. The *period* and *minPF* values used to find these items are 2 and 2, respectively. Let CI denote this sorted list of partial periodic items.

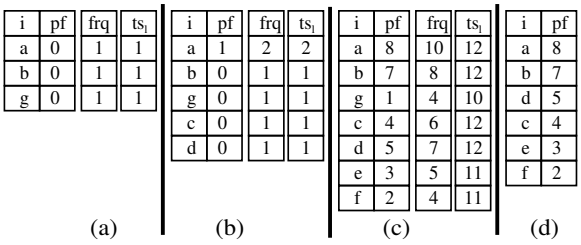

Figure 1: Construction of 3P-List. (a) After scanning the first transaction (b) After scanning the second transaction (c) After scanning the entire database (d) Final 3P-list containing the sorted list of items

After finding partial periodic items, we conduct another scan on the database and construct the prefix-tree of 3P-tree as in Algorithms 2 and 3. These algorithms are the same as those for constructing an FP-tree [5]. However, the major difference is that no node in 3P-tree maintains the $frequency$ as in an FP-tree. Figures 2 (a), (b)

Algorithm 1 Construction of 3P-List(TDB: temporal database, I: set of items, $minPF$: minimum periodic-frequency, per: period)

1: The *timestamps* of the last occurring transactions of all items in the 3P-list are explicitly recorded for each item in a temporary array, called ts_l. Similarly, the $frequency$ of all items in the 3P-list are explicitly recorded in another temporary array, called frq. These two arrays can be ignored after finding partial periodic items (or 1-itemsets).
2: Let $t = \{tid, ts_{cur}, X\}$ denote the current transaction with ts_{cur} and X representing the timestamp and an itemset, respectively.
3: **for** each transaction $t \in TDB$ **do**
4: **for** each item $i \in X$ **do**
5: **if** i does not exist in 3P-list **then**
6: Add i to the 3P-list and set $pf(i) = 0$, $frq(i) = 1$ and $ts_l(i) = ts_{cur}$.
7: **else**
8: **if** $ts_{cur} - ts_l(i) \leq per$ **then**
9: Set $pf(i)$ + +.
10: Set $ts_l(i) = ts_{cur}$ and $frq(i)$ + +.
11: Prune all aperiodic-items from the 3P-list that have pf less than $minPF$. Consider the remaining items in 3P-list as partial periodic items and sort them in descending order of their $frequency$. Let CI denote this sorted list of items.

Algorithm 2 3P-Tree(TDB, 3P-list)

1: Create the root of 3P-tree, T, and label it "*null*".
2: **for** each transaction $t \in TDB$ **do**
3: Set the timestamp of the corresponding transaction as ts_{cur}.
4: Select and sort the partial periodic items in t according to the order of CI. Let the sorted candidate item list in t be $[p|P]$, where p is the first item and P is the remaining list.
5: Call $insert\_tree([p|P], ts_{cur}, T)$.

Algorithm 3 insert_tree($[p|P]$, ts_{cur}, T)

1: **while** P is non-empty **do**
2: **if** T has a child N such that $p.itemName \neq N.itemName$ **then**
3: Create a new node N. Let its parent link be linked to T. Let its node-link be linked to nodes with the same itemName via the node-link structure. Remove p from P.
4: Add ts_{cur} to the leaf node.

and (c) show the 3P-tree constructed after scanning first, second and every transaction in the database, respectively. For simplicity, we do not show the node traversal pointers in trees; however, they are maintained like an FP-tree does.

4.2 Recursive Mining of 3P-tree

The 3P-tree is mined as follows. Start from each partial periodic item (as an initial suffix itemset), construct its conditional pattern base (a subdatabase, which consists of the set of prefix paths in the 3P-tree co-occurring with the suffix itemset), then construct its (conditional) 3P-tree, and perform mining recursively on such a tree. The pattern growth is achieved by the concatenation of the suffix itemset with the partial periodic itemsets generated from a conditional 3P-tree. The procedure to discover partial periodic itemsets from 3P-tree is shown in Algorithms 4 and 5. Mining of 3P-tree in Figure 2(c) is shown in Table 2.

Algorithm 4 3P-growth($Tree$, α)

1: **for** each a_i in the header of Tree **do**
2: Generate pattern $\beta = a_i \cup \alpha$. Collect all of the $a_i's$ ts-lists into a temporary array, TS^{β}, and calculate $PF(\beta)$ by calling $calculatePeriodicFrequency(TS^{\beta})$.
3: **if** $PF\beta \geq minPF$ **then**
4: Construct β's conditional pattern base then β's conditional 3P-tree $Tree_{\beta}$.
5: **if** $Tree_{\beta} \neq \emptyset$ **then**
6: call 3P-growth($Tree_{\beta}$, β);
7: Remove a_i from the $Tree$ and push the a_i's ts-list to its parent nodes.

Algorithm 5 calculatePeriodicFrequency(TS^{β}: list of timestamps containing β in TDB)

1: Set $PF(\beta) = 0$.
2: **for** *int* $i = 0$; $i < TS^{\beta}.length - 1$;+ + i **do**
3: **if** $TS^{\beta}[i+1] - TS^{\beta}[i] \geq per$ **then**
4: + + $PF(\beta)$.
5: return $PF(\beta)$.

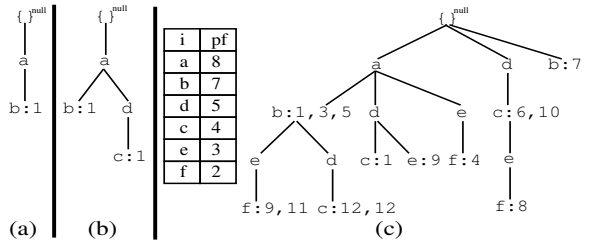

Figure 2: Construction of 3P-tree. (a) After scanning the first transaction (b) After scanning the second transaction and (c) Final 3P-tree generated after scanning the entire database

5 EXPERIMENTAL RESULTS

Since there exists no algorithm to find partial periodic itemsets in temporal databases, we only evaluate the proposed algorithm and show that our algorithm is memory and runtime efficient. Finally, we discuss the usefulness of the proposed model by demonstrating that approximately 44% of the event keywords found by a supervised event detection system [12] in Twitter data can also be discovered as partial periodic itemsets.

The 3P-growth algorithm was written in GNU C++ and run on a 2.66 GHz machine having 16 GB of memory. Ubuntu 14.04 is the operating system of our machine. The event detection system is written in python and java, and available for download at [11]. The experiments have been conducted using both synthetic

Table 2: Mining the 3P-tree by creating conditional (sub-)pattern bases

item	Conditional Pattern Base	Conditional 3P-tree	Partial periodic itemsets
f	$\{abe:9,11\}, \{ae:4\}, \{dce:8\}$	$\langle e:4,8,9,11\rangle$	$\{fe:2\}$
e	$\{ab:9,11\}, \{a:4\}, \{ad:9\}\{dc:8\}$	{a:9,9,11}	$\{ea:2\}$
c	$\{abd:12,12\}, \{ad:1\}, \{d:6,8,10\}$	$\langle d:1,6,8,10,12,12\rangle$	$\{cd:4\}$
d	$\{ab:12,12\}, \{a:1,9\}$	–	–
b	$\{a:1,3,5,9,11,12,12\}$	$\langle a:1,3,5,9,11,12,12\rangle$	$\{ab:5\}$

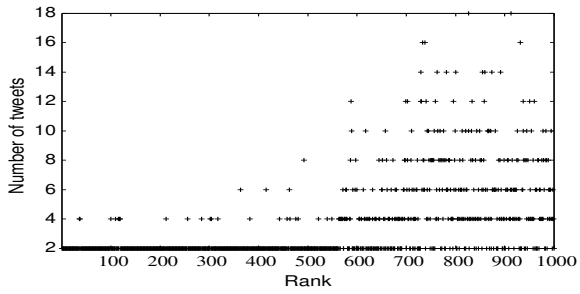

Figure 3: Number of tweets occurring at particular timestamp in Twitter database

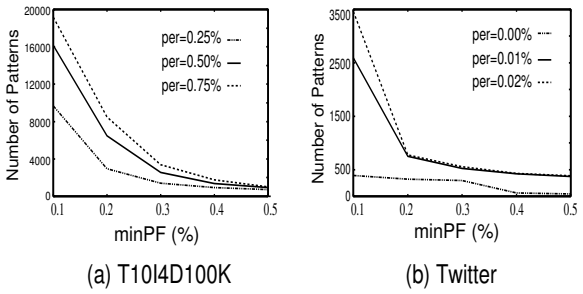

Figure 4: The number of partial periodic itemsets generated at different *per* and *minPF* values

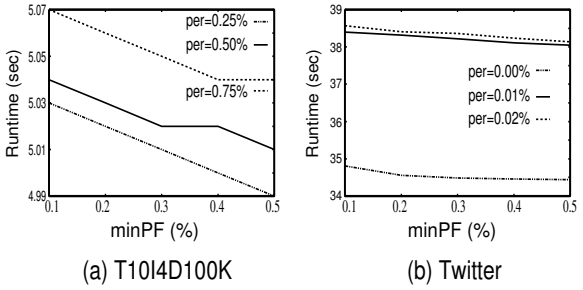

Figure 5: The runtime requirements of 3P-growth at different *per* and *minPF* values

(**T10I4D100K**) and real-world (**Twitter**) databases. The synthetic database, **T10I4D100K**, is generated by using the IBM data generator [1]. This database contains 870 items with 100,000 transactions. The Twitter database constitutes of 2,649,438 tweets collected from 10-march-2011 to 31-march-2011. These tweets are related to GEJE. The stop words have been removed from the tweets for computational reasons. As Twitter data is noisy, we have also removed highly infrequent words having $frequency$ less than 0.01%.

The transactions in T10I4D100K database do not have timestamps. Therefore, by considering that all transactions in this database are occurring at a fixed time interval, we have assigned timestamps for each transaction as increments of 1. That is, first transaction was assigned with the timestamp 1, the second transaction was assigned with the timestamp 2, and so on.

Figure 3 shows the number of tweets occurring at a particular timestamp in Twitter database. The X-axis represents the ranking of timestamps in ascending order, while the Y-axis represents the number of tweets occurring at a particular timestamp. For brevity, we have presented the results only for the first 1000 tweets in our Twitter database. It can be observed from this figure that many transactions share a common timestamp. Thus, twitter data represents a temporal database.

Figures 4(a) and (b) show the number of partial periodic itemsets generated at different *per* and *minPF* values in T10I4D100K and Twitter databases, respectively. The following two observations can be drawn from these figures: (*i*) Increase in *per* value may increase the number of partial periodic itemsets. The reason is that as *per* increases, the period threshold value of an itemset increases. (*ii*) Increase in *minPF* may decrease the number of partial periodic itemsets. The reason is increase in *minPF* increases the minimum number of cyclic repetitions necessary for a itemset to be a partial periodic itemset.

Figure 5(a) and (b) show the runtime requirements of 3P-growth at different *per* and *minPF* values in T10I4D100K and Twitter databases, respectively. It can be observed that changes in *per* and *minPF* values show the similar effect on runtime as of the generation of partial periodic itemsets. It is clear from these figures that 3P-growth algorithm can efficiently find partial periodic itemsets in very large databases even at low *minPF* and *per* values.

5.1 A case study: evaluation of partial periodic itemsets discovered in Twitter data

While investigating the usefulness of partial periodic itemsets discovered in Twitter data, we have observed that many generated partial periodic 1-itemsets (and their associations) were interesting as they were referring to the event GEJE. This motivated us to study the following: (*i*) Do event keywords in Twitter exhibit periodic behavior? and (*ii*) If event keywords exhibit periodic behavior, then what would be their percentage? The significance of this study is that if we find many event keywords exhibiting periodic behavior,

Table 3: Some of the interesting partial periodic itemsets and tweets containing the itemsets

Itemset	*PF*	Sample tweet
{jishin,helpme,anpi,-hinan,tsunami,-earthquake}	15,767	twitter社より。統一のハッシュタグなどが発表になりました。情報の統合に協力しましょう。 #jishin: 地震一般に関する情報 #j_j_helpme :救助要請 #hinan :避難 #anpi :安否確認 #311care: 医療系被災者支援情報" #NOSG **(summary: users were tweeting the list of hashtags provided by Twitter for GEJE)**
{fukushima, tsunami,-control}	421	Fukushima nuke plant out of control? RT talks to nuclear expert from Hiroshima, Japan http://bit.ly/fzwDpi #news #japan #fukushima #tsunami
{austraila, america,-warning,massive,-pacific, coast}	279	RT @bbcbreaking: #Tsunami warning is widened to incl rest of Pacific coast, incl #Australia and #South America massive #earthquake in #Japan

then one can use the proposed model as an unsupervised learning technique to derive some prior knowledge about event keywords and their associations in Twitter data.

Ritter et al. [12] discussed a supervised learning model to discover event keywords from tweets. We use this model for our experiment. This model annotates tweets using natural language processing techniques, generates a model from the training set of tweets and uses the model to extract event keywords from the test set of tweets. As the authors have already trained their model to identify event keywords in tweets, we have simply provided our Twitter data as the test set and extracted event keywords. A total of 330 event keywords have been extracted from the Twitter data. When we compared these event keywords against the partial periodic 1-itemsets generated at $per = 10\%$ and $minPF = 0.04\%$, we found that 145 event keywords have been generated as partial periodic 1-itemsets. In other words, 43.93% ($=\frac{145\times100}{330}$) of keywords have exhibited periodic behavior in Twitter data. This clearly demonstrates that periodic itemset mining can be used to find prior knowledge about event keywords and their associations in Twitter data. We have also carried out similar experiments on other Twitter databases and found that over 40% of event keywords were being generated as partial periodic 1-itemsets. Unfortunately, we are unable to present all of these results due to page limitation. Table 3 lists some of the generated partial periodic itemsets and their associated tweets.

6 CONCLUSIONS AND FUTURE WORK

We have proposed a model to find partial periodic itemsets in temporal databases. A new measure, *periodic-frequency*, has been discussed to determine the periodic interestingness of an itemset in a database. The partial periodic itemsets satisfy the anti-monotonic property. A pattern-growth algorithm based on this property has also been presented to find partial periodic itemsets. Experimental results show that the proposed model can find useful information and that the algorithm is efficient.

Our study has been confined to mining partial periodic itemsets in a static temporal database. The method developed here can be extended to incremental mining of partial periodic itemsets in temporal databases. As part of future work, we would like to investigate alternative measures of *periodic-frequency* to satisfy user and/or application requirements.

ACKNOWLEDGMENTS

The authors would like to thank J.N. Venkatesh and Prof. P. Krishna Reddy for sharing their domain expertise in periodic pattern mining and providing very useful information regarding TWICAL software.

REFERENCES

[1] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining association rules between sets of items in large databases. In *SIGMOD*. 207–216.
[2] Komate Amphawan, Philippe Lenca, and Athasit Surarerks. 2009. Mining Top-K Periodic-Frequent Pattern from Transactional Databases without Support Threshold. In *Advances in Information Technology*. 18–29.
[3] Walid G. Aref, Mohamed G. Elfeky, and Ahmed K. Elmagarmid. 2004. Incremental, Online, and Merge Mining of Partial Periodic Patterns in Time-Series Databases. *IEEE TKDE* 16, 3 (2004), 332–342.
[4] Jiawei Han, Wan Gong, and Yiwen Yin. 1998. Mining Segment-Wise Periodic Patterns in Time-Related Databases.. In *KDD*. 214–218.
[5] Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. 2004. Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach. *Data Min. Knowl. Discov.* 8, 1 (Jan. 2004), 53–87.
[6] R. Uday Kiran and Masaru Kitsuregawa. 2014. Novel Techniques to Reduce Search Space in Periodic-Frequent Pattern Mining. In *DASFAA (2)*. 377–391.
[7] R. Uday Kiran, Masaru Kitsuregawa, and P. Krishna Reddy. 2016. Efficient discovery of periodic-frequent patterns in very large databases. *Journal of Systems and Software* 112 (2016), 110–121.
[8] R. Uday Kiran and P. Krishna Reddy. 2010. Towards Efficient Mining of Periodic-Frequent Patterns in Transactional Databases. In *DEXA (2)*. 194–208.
[9] R. Uday Kiran and P. Krishna Reddy. 2011. An Alternative Interestingness Measure for Mining Periodic-Frequent Patterns. In *DASFAA (1)*. 183–192.
[10] S. Ma and J.L. Hellerstein. 2001. Mining partially periodic event patterns with unknown periods. In *ICDE*. 205–214.
[11] Alan Ritter. 2012. TWICAL software. http://github.com/aritter/twitter_nlp. (2012). [Online; accessed 16-May-2017].
[12] Alan Ritter, Mausam, Oren Etzioni, and Sam Clark. 2012. Open Domain Event Extraction from Twitter. In *KDD*. 1104–1112.
[13] Akshat Surana, R. Uday Kiran, and P. Krishna Reddy. 2011. An Efficient Approach to Mine Periodic-Frequent Patterns in Transactional Databases. In *PAKDD Workshops*. 254–266.
[14] Pang-Ning Tan, Vipin Kumar, and Jaideep Srivastava. 2002. Selecting the Right Interestingness Measure for Association Patterns. In *Knowledge Discovery and Data Mining*. 32–41.
[15] Syed Khairuzzaman Tanbeer, Chowdhury Farhan Ahmed, Byeong Soo Jeong, and Young Koo Lee. 2009. Discovering Periodic-Frequent Patterns in Transactional Databases. In *PAKDD*. 242–253.
[16] J. N. Venkatesh, R. Uday Kiran, P. Krishna Reddy, and Masaru Kitsuregawa. 2016. Discovering Periodic-Frequent Patterns in Transactional Databases Using All-Confidence and Periodic-All-Confidence. In *DEXA*. 55–70.
[17] Jiong Yang, Wei Wang, and Philip S. Yu. 2003. Mining Asynchronous Periodic Patterns in Time Series Data. *IEEE Trans. Knowl. Data Eng.* (2003), 613–628.
[18] R. Yang, W. Wang, and P.S. Yu. 2002. InfoMiner+: mining partial periodic patterns with gap penalties. In *ICDM*. 725–728.
[19] Minghua Zhang, Ben Kao, David W. Cheung, and Kevin Y. Yip. 2007. Mining periodic patterns with gap requirement from sequences. *ACM Trans. Knowl. Discov. Data* 1, 2 (Aug. 2007).