Discovering Multiple Time Lags of Temporal Dependencies from Fluctuating Events

Wentao Wang(✉), Chunqiu Zeng, and Tao Li

School of Computing and Information Sciences, Florida International University, Miami, FL 33199, USA
{wwang041,czeng001,taoli}@cs.fiu.edu

Abstract. As one of the key features of temporal dependency, time lag plays an important role in analyzing sequential data and predicting the developing trend. Huge number of temporal mining approaches have been successfully applied in many applications, like finance, environmental science and health-care. However, these approaches cannot effectively deal with a more realistic scenario, where more than one types of time lags are existed in sequences and all of them are fluctuating due to the inevitable noise. In this paper, we study the problem of discovering multiple time lags of temporal dependencies from event sequences considering the randomness property of the hidden time lags. We first present a parametric model as well as an EM-based solution for solving this problem. Then two approximate approaches are proposed for efficiently finding diverse types of time lags without significant loss of accuracy. Extensive empirical studies on both synthetic and real datasets demonstrate the efficiency and effectiveness of our proposed approaches.

Keywords: Time lag · Temporal dependency · Event mining

1 Introduction

In the past several decades, temporal data mining has been widely applied in many domains, such as finance [8], computer science [19], environmental science [2]. The goal of temporal data mining is to discover hidden temporal dependencies, unexpected trends or other subtle relationships in sequential data [15,27]. As an important task in temporal data mining, temporal dependency discovery has been extensively studied for identifying hidden interactions and mining useful information from sequential data. Specifically, suppose A and B are two types of items, a temporal dependency for A and B, written as $A \to B$, could be discovered when the occurrence of B depends on the occurrence of A.

Traditional temporal mining methods either utilize some statistical techniques [18] or employ a predefined window [6] to discover temporal dependencies. The main drawback of these previous methods is that they cannot discover interleaved dependencies, since all of these methods are based on an assumption that every item A only has a dependency relation with its first following B. However, interleaved dependencies are very common in real application scenarios, where an item A could have a dependency relation with any following B. For example, as shown in Fig. 1, an event *High CPU Utilization Alert* can be triggered by an event *Abnormal Process* in system management domain. Since sometimes one abnormal process may be solved very quickly after it appeared, it would not trigger any *High CPU Utilization Alert*. Hence, this abnormal process (at time point 38) does not have the corresponding *High CPU Utilization Alert*. Two well-designed algorithms are presented in [28] for mining interleaved temporal dependencies from sequential data with satisfactory time cost and space cost.

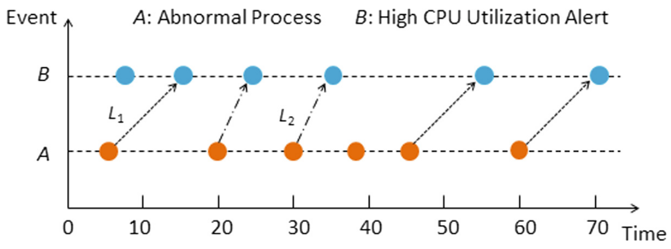


Fig. 1. Two types of time lags for temporal dependency *Abnormal Process* $\rightarrow$ *High CPU Utilization Alert*.

Time lag, one of the key features of temporal dependency, plays an essential role in interpreting the cause of discovered temporal dependencies and predicting the evolving trends for future data. Existing work related to time lag discovery suppose the time lag between two correlated events is constant and fluctuations can be ignored [9,28]. However, because of fluctuation, noise and missing data, there are more than one types of time lags are existed in event sequences in the real application scenarios, and each of them involves randomness property.

As summarized in [30], taking randomness of time lag into consideration in temporal dependencies discovery is a big challenge, since (1) the number of time lag candidates in large sequential datasets are tremendous, and (2) the hidden time lags may oscillate with noise formed in data collection process. The model proposed in [30] assumes the distribution of time lags follows a normal distribution $\mathcal{N}(\mu, \sigma^2)$. Nevertheless, in practice, we find that the interleaved time lags often follow multiple distributions in one temporal dependency rather than a single normal distribution. As shown in Fig. 1, there are two types of time lags, i.e., L_1 and L_2, between event *Abnormal Process* and *High CPU Utilization Alert*. The reason caused this situation is that event *Abnormal Process* represents many different kinds of abnormal processes, and each of them may have different effects on the system with respect to CPU utilization.

In order to overcome the limitations of existing approaches and deal with real application scenarios better, in this paper, we study the problem of mining multiple time lags with randomness property for temporal dependencies. The contribution of this paper is summarized as follows:

- Investigates the problem of discovering multiple types of fluctuating interleaved time lags, and proposes a parametric model to formulate the randomness of time lags for temporal dependencies between pairwise events.
- Presents an EM-based solution for mining multiple types of time lags. Moreover, for efficiently mining diverse time lags from large event sequences, two approximate algorithms are proposed with satisfactory performance.
- Conducts extensive experiments on different synthetic datasets and several real datasets. The experimental results demonstrate that all our proposed algorithms could find multiple types of time lags effectively.

The rest of the paper is organized as follows. Section 2 summarizes the existing work for temporal data mining. We formulate the problem for discovering multiple time lags from fluctuating events in Sect. 3. A parametric model as well as an EM-based solution are presented in Sect. 4. Section 5 presents two approximate algorithms with better efficiency. Extensive experimental results are reported in Sect. 6. Finally, we conclude this paper in Sect. 7.

2 Related Work

Temporal dependency discovery approaches have been extensively applied in numerous real applications with various dataset types. Transaction data, commonly known as market basket transactions [27], is a collection of transactions, in which each transaction contains a set of items. Transaction data arises in many business related applications, including marketing promotions, advertisements and recommendation systems. Discovering temporal dependencies from transaction data is equivalent to finding frequent itemsets satisfying some predefined thresholds. Many algorithms are proposed for efficiently mining frequent itemsets from transaction datasets, such as GSP [26], FreeSpan [10], and PrefixSpan [22].

Mining temporal dependencies from time series data has been recognized as one of the key tasks in time series analysis [29]. For time series data, each record is a series of measurements taken over time. A temporal dependency, often called causal relationship, among time series can be seen as a correlation on multiple time series, which states one time series is significantly helpful to predict the future trend of another time series [7,20]. In particular, if time series A causes time series B, then the prediction of future value of B can be improved by utilizing A and B together. In recent years, the problem of identifying causal relationship between various time series has attracted widespread attention, and two effective frameworks has became very popular in temporal dependency inference, i.e., Dynamic Bayesian Network [13,25] and Granger Causality [3,4].

Event data, converted from textual logs which generated by modern computing systems, has been widely used in system and network management related applications [11]. Differing from time series where the value of data item is continuous, event data denotes the discrete data item values [16]. An event sequence is an ordered finite sequence, in which each element is a tuple consisting of one instance of some event and its corresponding timestamp. A lot of research on event mining are proposed for discovering relationships of events [21,23]. Our work also focuses on event data, where only the timestamps of items are available and no other information can be utilized to find temporal dependencies.

Unlike previous work which can only discover fixed time lags, in this paper, we proposed a parametric model to extract the probability distributions of time lags. Considering probability distributions could precisely depict the randomness property of time lags, our method provides more flexibility and usability than fixed ones.

3 Problem Definition and Formulation

3.1 Problem Definition

Let $\mathbf{\Omega}$ be the event space comprises all possible events. An event sequence $\mathbf{S}$ over $\mathbf{\Omega}$ is a finite ordered list with the form $\mathbf{S} = e_1 e_2 \dots e_u$. Every element $e_i \in \mathbf{S}$ is a tuple $e_i = (E_i, t_i)$ indicating an instance of event $E_i \in \mathbf{\Omega}$ occurred at time t_i.

Assume A be a type of event coming from event space $\mathbf{\Omega}$, $\mathbf{S_A}$ be a subsequence of $\mathbf{S}$ which only consists of instances of event A. Because all elements in $\mathbf{S_A}$ belong to the same type of event, we simplify $\mathbf{S_A}$ as a sequence of timestamp, i.e., $\mathbf{S_A} = a_1 a_2 \dots a_m$, where a_i is the i^{th} timestamp of event A's instances. Similarly, for another type of event B, we denote $\mathbf{S_B} = b_1 b_2 \dots b_n$.

If there is a temporal dependency $A \rightarrow_L B$, for any associated timestamp pair a_i and b_j, there always exists a relation $b_j = a_i + L$ indicating an event A occurred at a_i is followed by an event B occurred at b_j after a time lag L.

Theoretically, the time lag L should be a constant. However, the noise is inevitable during data collection process because of various factors, such as missing records, incorrect values and recording delay. In order to discover underlying temporal dependencies effectively, in this paper, the time lag L is defined as

$$L = \mu + \epsilon \tag{1}$$

where μ is a constant representing the true time lag and ϵ is a random variable indicating the noise. Hence, time lag L is a random variable.

In our practice, we find that the time lag L often follows a more complicated distribution rather than a single normal distribution supposed in previous work [30]. Without loss of generality, we assume this complicated distribution is consisted of K different probability distributions, and we want to discover multiple time lags following various probability distributions from datasets. Definition 1 provides the formal description of the problem we studied in this paper.

Definition 1. *For two event types A and B, suppose there are K different temporal dependencies $A \rightarrow_{L_1} B, A \rightarrow_{L_2} B, \dots, A \rightarrow_{L_K} B$ existed in a given dataset, our goal is to learn a time lag L, which is consisted of K types of time lags $L_1, L_2, \dots, L_K$ following K different probability distributions, respectively.*

3.2 Problem Formulation

Given two timestamp sequences $\mathbf{S_A}$ and $\mathbf{S_B}$, suppose the distribution of time lag L is determined by the parameters $\mathbf{\Theta}$, which are independent from the occurrences of event A. Therefore, the problem of discovering the temporal dependency $A \rightarrow_L B$ is equivalent to learning the distribution of time lag L through the maximum likelihood parameters $\mathbf{\Theta}$ defined by

$$\hat{\mathbf{\Theta}} = \arg\max_{\mathbf{\Theta}} P(\mathbf{\Theta}|\mathbf{S_A}, \mathbf{S_B}). \tag{2}$$

Applying Bayes' theorem to Eq. (2), we have

$$\ln P(\mathbf{\Theta}|\mathbf{S_A}, \mathbf{S_B}) = \ln P(\mathbf{S_B}|\mathbf{S_A}, \mathbf{\Theta}) + \ln P(\mathbf{S_A}) + \ln P(\mathbf{\Theta}) - \ln P(\mathbf{S_A}, \mathbf{S_B}). \tag{3}$$

In Eq. (3), only $\ln P(\mathbf{S_B}|\mathbf{S_A}, \mathbf{\Theta})$ and $\ln P(\mathbf{\Theta})$ are related to the parameters $\mathbf{\Theta}$. Therefore, the problem of learning time lag L can be simplified into the problem of solving the following equation

$$\hat{\mathbf{\Theta}} = \arg\max_{\mathbf{\Theta}} \ln P(\mathbf{S_B}|\mathbf{S_A}, \mathbf{\Theta}). \tag{4}$$

4 Modeling and Solution

4.1 Time Lag Modeling

For a temporal dependency $A \rightarrow_L B$, we assume that every occurrence of event B is only determined by event A and time lag L, i.e., every occurrence of event B is mutually independent with each other. Therefore,

$$P(\mathbf{S_B}|\mathbf{S_A}, \mathbf{\Theta}) = \prod_{j=1}^{n} P(b_j|\mathbf{S_A}, \mathbf{\Theta}). \tag{5}$$

For every timestamp b_j, a latent variable z_{ijk} is introduced to model the relation between b_j and one timestamp of event A, denoted as a_i. Specifically,

$$z_{ijk} = \begin{cases} 1, & \textit{the } i^{th} \textit{ event } A \textit{ implies the } j^{th} \textit{ event } B \textit{ following } k^{th} \textit{ distribution;} \\ 0, & \textit{otherwise.} \end{cases} \tag{6}$$

Hence, the relation between b_j and sequence $\mathbf{S_A}$ can be represented by a latent matrix $\mathbf{Z_j} = \{z_{ijk}\}_{m \times K}$, in which only one element equals to 1 and all other elements are 0. If $z_{ijk} = 1$, then cell (i, k) in $\mathbf{Z_j}$ equals 1.

Based on the definition of latent variable z_{ijk}, the distribution of latent matrix $\mathbf{Z_j}$ and the conditional distribution of b_j given $\mathbf{Z_j}$ are shown as:

$$P(\mathbf{Z_j}) = \prod_{i=1}^{m} \prod_{k=1}^{K} P(z_{ijk} = 1)^{z_{ijk}}. \tag{7}$$

$$P(b_j|\mathbf{Z_j}, \mathbf{S_A}, \mathbf{\Theta}) = \prod_{i=1}^{m}\prod_{k=1}^{K} P(b_j|z_{ijk} = 1, a_i, \mathbf{\Theta})^{z_{ijk}}. \tag{8}$$

Therefore, the joint distribution $P(b_j, \mathbf{Z_j}|\mathbf{S_A}, \mathbf{\Theta})$ can be written as follow:

$$P(b_j, \mathbf{Z_j}|\mathbf{S_A}, \mathbf{\Theta}) = \prod_{i=1}^{m}\prod_{k=1}^{K} \{P(b_j|z_{ijk} = 1, a_i, \mathbf{\Theta}) \times P(z_{ijk} = 1)\}^{z_{ijk}}. \tag{9}$$

So the marginal distribution of b_j is obtained by

$$P(b_j|\mathbf{S_A}, \mathbf{\Theta}) = \sum_{\mathbf{Z_j}} P(b_j, \mathbf{Z_j}|\mathbf{S_A}, \mathbf{\Theta}) = \sum_{i=1}^{m}\sum_{k=1}^{K} P(b_j|z_{ijk} = 1, a_i, \mathbf{\Theta}) \times P(z_{ijk} = 1). \tag{10}$$

Combining Eqs. (4), (5) and (10) together, the log-likelihood function can be rewritten as:

$$\ln P(\mathbf{S_B}|\mathbf{S_A}, \mathbf{\Theta}) = \sum_{j=1}^{n} \ln \sum_{i=1}^{m}\sum_{k=1}^{K} P(b_j|z_{ijk} = 1, a_i, \mathbf{\Theta}) \times P(z_{ijk} = 1). \tag{11}$$

For simplicity, let $\pi_{ijk} = P(z_{ijk} = 1)$, where $0 \leq \pi_{ijk} \leq 1$, $\sum_{i=1}^{m}\sum_{k=1}^{K} \pi_{ijk} = 1$.

Time lag L is consisted of K different time lags $L_1, L_2, \ldots, L_K$ with K different probability distributions. For any one temporal dependency $A \rightarrow_{L_t} B$ $(1 \leq t \leq K)$, we have $L_t = \mu_t + \epsilon_t$. Based on the Central Limit Theorem, we assume that noise ϵ follows the normal distribution with zero-mean value, i.e., $\epsilon_t \sim \mathcal{N}(0, \sigma_t^2)$, where σ_t^2 represents the variance of current distribution. Since μ_t is a constant, the distribution of L_t can be expressed as $L_t \sim \mathcal{N}(\mu_t, \sigma_t^2)$. Therefore, time lag L can be regard as a mixture of K different normal distributions with various μ and σ^2. Hence, if $z_{ijk} = 1$, then

$$P(b_j|z_{ijk} = 1, a_i, \mathbf{\Theta}) = P(b_j|a_i, \mu_k, \sigma_k^2) = \mathcal{N}(b_j - a_i|\mu_k, \sigma_k^2). \tag{12}$$

Consequently, Eq. (11) can be expressed as follow:

$$\ln P(\mathbf{S_B}|\mathbf{S_A}, \mathbf{\Theta}) = \sum_{j=1}^{n} \ln \sum_{i=1}^{m}\sum_{k=1}^{K} \pi_{ijk} \times \mathcal{N}(b_j - a_i|\mu_k, \sigma_k^2). \tag{13}$$

Based on Eq. (13), the problem described in Eq. (4) is equivalent to the following equation

$$\begin{aligned}(\hat{\mu}_k, \hat{\sigma}_k^2) = &\underset{\substack{\mu_k, \sigma_k^2 \\ k \in \{1,\ldots,K\}}}{\arg\max} \sum_{j=1}^{n} \ln \sum_{i=1}^{m}\sum_{k=1}^{K} \pi_{ijk} \times \mathcal{N}(b_j - a_i|\mu_k, \sigma_k^2) \\ &\textbf{s.t. } \textit{for every } j \in [1, n], \ \sum_{i=1}^{m}\sum_{k=1}^{K} \pi_{ijk} = 1.\end{aligned} \tag{14}$$

4.2 Maximization

Equation (14) can be solved by expectation-maximization (EM) algorithm [5], since it is one kind of mixture model. For applying EM algorithm, consider the expected log likelihood function of complete data $\{\mathbf{S_B}, \mathbf{\Theta}\}$ at first. Suppose parameters $\mathbf{\Theta}$ is already known and $\mathbf{Z} = \{z_{ijk}\}_{m\times n\times K}$ is a latent matrix. Then, based on Eqs. (7) and (8), we have

$$P(\mathbf{S_B}, \mathbf{Z}|\mathbf{S_A}, \mathbf{\Theta}) = \prod_{j=1}^{n} P(b_j|\mathbf{Z_j}, \mathbf{S_A}, \mathbf{\Theta}) \times P(\mathbf{Z_j}) = \prod_{i=1}^{m}\prod_{j=1}^{n}\prod_{k=1}^{K} \{\mathcal{N}(b_j - a_i|\mu_k, \sigma_k^2) \times \pi_{ijk}\}^{z_{ijk}}. \tag{15}$$

Therefore, the expectation can be expressed as

$$\mathcal{Q}(\mathbf{\Theta}, \mathbf{\Theta}') \triangleq \mathbb{E}[\ln P(\mathbf{S_B}, \mathbf{Z}|\mathbf{S_A}, \mathbf{\Theta})] = \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{K} \mathbb{E}[z_{ijk}] \times \left\{\ln \mathcal{N}(b_j - a_i|\mu_k, \sigma_k^2) + \ln \pi_{ijk}\right\}. \tag{16}$$

where $\mathbf{\Theta}'$ is the parameters estimated on the previous iteration. Using r_{ijk} to denote $\mathbb{E}[z_{ijk}]$, i.e.,

$$r_{ijk} \triangleq \mathbb{E}[z_{ijk}] = P(z_{ijk} = 1|\mathbf{S_A}, \mathbf{S_B}, \mathbf{\Theta}') = \frac{\pi'_{ijk} \times \mathcal{N}(b_j - a_i|\mu'_k, \sigma'^2_k)}{\sum_{i=1}^{m}\sum_{k=1}^{K} \pi'_{ijk} \times \mathcal{N}(b_j - a_i|\mu'_k, \sigma'^2_k)}. \tag{17}$$

Then Eq. (16) can be rewritten as

$$\mathcal{Q}(\mathbf{\Theta}, \mathbf{\Theta}') = \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{K} r_{ijk} \times \left\{\ln \mathcal{N}(b_j - a_i|\mu_k, \sigma_k^2) + \ln \pi_{ijk}\right\}. \tag{18}$$

The parameters μ_k, σ_k^2 and π_{ijk} can be learned by maximizing $\mathcal{Q}(\mathbf{\Theta}, \mathbf{\Theta}')$.

$$\mu_k = \frac{1}{N_k}\sum_{i=1}^{m}\sum_{j=1}^{n} r_{ijk}(b_j - a_i) \tag{19}$$

$$\sigma_k^2 = \frac{1}{N_k}\sum_{i=1}^{m}\sum_{j=1}^{n} r_{ijk}(b_j - a_i - \mu_k)^2 \tag{20}$$

$$\pi_{ijk} = \frac{1}{n}\sum_{j=1}^{n} r_{ijk} \tag{21}$$

where $N_k = \sum_{i=1}^{m}\sum_{j=1}^{n} r_{ijk}$.

Using this EM-based algorithm, called *EMLag* algorithm, we can find the maximum likelihood estimates of parameters $\mathbf{\Theta}$. Algorithm 1 states the pseudo-code of *EMLag* algorithm with the time complexity $\mathcal{O}(rmnK)$, where m and n are the number of timestamps of event A and B, respectively, K is the number of distributions and r indicates iteration number.

Algorithm 1. The EMLag algorithm

```
1: procedure EMLAG($\mathbf{S_A}$, $\mathbf{S_B}$)                                  ▷ $|\mathbf{S_A}|=m$, $|\mathbf{S_B}|=n$
2:     Initialize $r'_{ijk} = \frac{1}{mK}$, choose $\mu'_k$ and $\sigma'^2_k$ randomly.   ▷ Initialization
3:     while TRUE do
4:         Evaluate $r_{ijk}$ by Eq. (17).                                            ▷ Expectation
5:         Update $\mu_k$ and $\sigma^2_k$ by Eqs. (19) and (20), respectively.    ▷ Maximization
6:         if parameters converge then                                           ▷ Convergence test
7:             return $\mu_k$ and $\sigma^2_k$                                  ▷ $k = 1, 2, \ldots, K$
8:         end if
9:     end while
10: end procedure
```

5 Time Lag Discovery

Based on *EMLag* algorithm, we design two approximate algorithms for mining multiple time lags from large datasets more efficiently. Both of these two algorithms could achieve good efficiency without significant loss of accuracy.

5.1 winEMLag Algorithm

Intuitively, suppose a temporal dependency $A \rightarrow B$ is exist, timestamp b_j is more likely to be implied by timestamp a_i if the index i is close to the index j rather than far from j. Hence, for mining various temporal dependencies from large event sequences efficiently, for every b_j, we only select a subset of event A's timestamps whose index is close to j for calculation instead of all of them.

Algorithm 2. The winEMLag algorithm

```
1: procedure EXPECTATION($\mathbf{S_A}$, $\mathbf{S_B}$, $\lambda$)          ▷ $\lambda$ is predefined, $0 < \lambda \leq 1$
2:     $l = \lambda \times m$                                  ▷ $l$ is the window length, $|\mathbf{S_A}| = m$
3:     $left = 0, right = 0$                                                  ▷ Index bound
4:     for each $b_j$ do
5:         if $j - l/2 \geq 0$ and $j + l/2 \leq m - 2$ then
6:             $left = j - l/2$, $right = left + l - 1$
7:         else
8:             if $j - l/2 \geq 0$ then
9:                 $right = m - 1$, $left = m - l$
10:            else
11:                $left = 0$, $right = l - 1$
12:            end if
13:        end if
14:        Select $a_i$ into $w_j$ where $i \in [left, right]$.
15:        Evaluate $r_{ijk}$ utilizing set $w_j$.
16:    end for
17: end procedure
```

Let w_j be a subset of A's timestamps used to estimate the relation between event A and b_j. Our goal is to fill in w_j so that, compared with remainders, the indexes of A's timestamps in w_j are much closer to j. Inspired by Sliding Window Model [1], we design an approximate algorithm *winEMLag* for speeding up the mining process of *EMLag* algorithm. Algorithm 2 shows the Expectation procedure of *winEMLag* algorithm. In each iteration, the update operations in Maximization procedure of *winEMLag* algorithm will also utilize each subset w_j.

In *winEMLag* algorithm, parameter λ is a user-specified parameter representing the ratio between the length of window l and the length of event sequence. Therefore, the length of window l can be calculated by $l = \lambda \times |\mathbf{S_A}|$. Since the size of each w_j in *winEMLag* algorithm is much smaller than the size of sequence $\mathbf{S_A}$ in *EMLag* algorithm, *winEMLag* algorithm could achieve better efficiency.

5.2 appEMLag Algorithm

During each iteration of *EMLag* algorithm, we find that, for every specific distribution k, the responsibility r_{ijk} describing the likelihood that the i^{th} event A implies the j^{th} event B following k^{th} normal distribution, becomes smaller with the deviation of $b_j - a_i$ from the estimated time lag μ_k increasing. In other words, r_{ijk} will close to 0 as $|b_j - a_i - \mu_k|$ becomes larger. Based on this observation, we design an approximate algorithm for efficiently estimating parameters μ_k and σ_k^2 by ignoring those $r_{ijk}(b_j - a_i)$ and $r_{ijk}(b_j - a_i - \mu_k)^2$ with small r_{ijk} in both Eqs. (19) and (20). Since in real application scenarios, the time spans of given event sequences are very long, and most r_{ijk} are very small, the loss of accuracy of this approximate method is acceptable.

We introduce two parameters ϵ and δ to help distinguishing retained part and neglected part of r_{ijk}. Given b_j, let ϵ_j be the sum of the responsibility r_{ijk} which will be neglected, i.e., $\epsilon_j = \sum_{k=1}^{K} \sum_{\{i|a_i\ is\ neglected\}} r_{ijk}$, and ϵ be the largest one among all the ϵ_j, that is, $\epsilon = \max_{1 \leq j \leq n} \{\epsilon_j\}$. In practice, parameter ϵ can be predefined by users based on the application scenario.

Recall that in Eqs. (19) and (20), each pair of μ_k and σ_k are calculated by their corresponding r_{ijk}. Therefore, for every b_j, we suppose set C_{jk} includes all retained r_{ijk} which will be used to estimate μ_k and σ_k. Since all timestamps of event A are consecutive in ascending order, the index i for timestamps of event A in set C_{jk} are also consecutive. Let δ_{jk} be the ratio of the sum of retained r_{ijk} in set C_{jk} to the sum of all retained r_{ijk} for the given b_j, and hence δ is a $n \times K$ matrix filled in by all δ_{jk}. In each iteration, given b_j, δ_{jk} can be updated by the following equations:

$$AVG_{jk} \triangleq \frac{\sum_{\{r_{ijk} \in C_{jk}\}} r_{ijk}}{|C_{jk}|}, \ \delta_{jk} = \frac{AVG_{jk}}{\sum_{k=1}^{K} AVG_{jk}} \tag{22}$$

To guarantee the sum of neglected r_{ijk} is less than ϵ, for every b_j, the sum of retained r_{ijk} should be greater than $1 - \epsilon$, i.e., $\sum_{k=1}^{K} \delta_{jk} \geq 1 - \epsilon$. In order to minimize the size of C_{jk}, we adopt a greedy way to select timestamps a_i from all timestamps of event A. Specifically, given b_j and distribution k, we add a_i

into C_{jk} with its corresponding r_{ijk} in decreasing order until the summation of r_{ijk} in C_{jk} is greater than δ_{jk}. Algorithm 3 describes how to find the minimum and maximum indexes of a_i in C_{jk}.

Algorithm 3. The greedyBound algorithm

```
1: procedure GREEDYBOUND($\mathbf{S_A}$, $b_j$, $\mu'_k$, $\delta'_{jk}$, $\epsilon$)
2:     $t = b_j - \mu'_k$
3:     Locate the closest $a_i$ to $t$ using binary search.
4:     $min_{jk} = i$, $max_{jk} = i$
5:     $prob = 0.0$
6:     while $prob < \delta'_{jk} \times (1 - \epsilon)$ do
7:         if $r_{(min_j - 1)jk} \geq r_{(max_j + 1)jk}$ then
8:             $i = min_{jk} - 1$
9:             $min_{jk} = i$
10:        else
11:            $i = max_{jk} + 1$
12:            $max_{jk} = i$
13:        end if
14:        Add $a_i$ to $C_{jk}$.
15:        $prob = prob + r_{ijk}$
16:    end while
17:    return $min_{jk}$ and $max_{jk}$
18: end procedure
```

Based on *greedyBound* algorithm, we present an approximate algorithm, called *appEMLag* algorithm, to efficiently estimate parameters μ_k and σ_k^2 without significant loss of accuracy. As described in Algorithm 4, the time cost of

Algorithm 4. The appEMLag algorithm

```
1: procedure APPEMLAG($\mathbf{S_A}$, $\mathbf{S_B}$, $\epsilon$)            ▷ $\epsilon$ is predefined, $0 < \epsilon \leq 1$
2:     Initialize $\delta'_{jk} = \frac{1}{K}$, choose $\mu'_k$ and $\sigma'^2_k$ randomly.     ▷ Initialization
3:     while TRUE do
4:         for each $b_j$ do
5:             for $k \leftarrow 1, K$ do
6:                 Get $min_{jk}$, $max_{jk}$ by greedyBound.   ▷ Find the index bound of $a_i$
7:             end for
8:             Evaluate $r_{ijk}$ utilizing sets $C_{j1}, \ldots, C_{jK}$.      ▷ Expectation
9:         end for
10:        Update $\mu_k$ and $\sigma_k^2$ by Eqs. (19) and (20) within the     ▷ Maximization
           bound, respectively, and update $\delta_{jk}$ by Eq. (22).
11:        if parameters converge then                      ▷ Convergence test
12:            return $\mu_k$ and $\sigma_k^2$                        ▷ $k = 1, 2, \ldots, K$
13:        end if
14:    end while
15: end procedure
```

appEMLag algorithm is $\mathcal{O}(rnK(\log m + t))$, where m and n are the number of timestamps of event A and B, respectively, r indicates iteration number, K is the number of distributions, and t is the average size of all C_{jk}. Since $t \ll m$ and $\log m \ll m$, *appEMLag* algorithm is much faster than *EMLag* algorithm.

6 Empirical Study

This section presents empirical studies of our proposed algorithms on both synthetic datasets and real datasets with respect to effectiveness and efficiency. To demonstrate the performance of our proposed algorithms, we implement all of them using Java 1.7, and execute them on a computer with Linux 2.6.32. This computer is equipped with Intel(R) Xeon(R) CPU with 24 cores running at speed of 2.50 GHz, and the total memory of it is 126G.

6.1 Synthetic Data

Synthetic Data Generation. In our experiments, we execute our proposed algorithms on five different synthetic datasets. Parameters shown in Table 1 are utilized to generate synthetic datasets. Each dataset consists of two event sequences $\mathbf{S_A}$ and $\mathbf{S_B}$ with same length, and we assume there are two types of temporal dependency exist in each dataset. That is to say, $K = 2$. Moreover, we use the exponential distribution to simulate the inter-arrival time between two adjacent events [17]. The way of generating $\mathbf{S_A}$ and $\mathbf{S_B}$ is shown below.

Table 1. Parameters used for generating synthetic data

Name	Description
N	The number of events in one synthetic event sequence
K	The number of types of time lag
β_{min}	The minimum value for the average inter-arrival time β
β_{max}	The maximum value for the average inter-arrival time β
μ_{min}	The minimum value for the true time lag μ
μ_{max}	The maximum value for the true time lag μ
σ^2_{min}	The minimum value for the variance of time lag
σ^2_{max}	The maximum value for the variance of time lag

1. Randomly choose parameters β from $[\beta_{min}, \beta_{max}]$, μ_1 and μ_2 from $[\mu_{min}, \mu_{max}]$ and σ_1^2 and σ_2^2 from $[\sigma^2_{min}, \sigma^2_{max}]$, respectively.
2. Generate $N/2$ timestamps for event A, where the inter-arrival time between two neighbors follows the exponential distribution with parameter β.
3. For each timestamp a_i of event A, the time lag is randomly generated according to normal distribution with parameters μ_1 and σ_1^2.

4. Combine all the timestamps associated with their types to form two event sequence $\mathbf{S_{A_1}}$ and $\mathbf{S_{B_1}}$.
5. Repeat Steps 2–4 to generate another two event sequences $\mathbf{S_{A_2}}$ and $\mathbf{S_{B_2}}$ with parameters μ_2 and σ_2^2 chosen in Step 1.
6. Merge $\mathbf{S_{A_1}}$ and $\mathbf{S_{A_2}}$ to form the sequence $\mathbf{S_A}$ with timestamps in ascending order, then merge $\mathbf{S_{B_1}}$ and $\mathbf{S_{B_2}}$ to form the sequence $\mathbf{S_B}$ based on the indexes of their corresponding a_i.

In our experiments, we set $\beta_{min} = 5$, $\beta_{max} = 50$, $\mu_{min} = 25$, $\mu_{max} = 100$, $\sigma^2_{min} = 5$ and $\sigma^2_{max} = 400$ to generate five synthetic datasets with different parameter N. The number of events in sequence $\mathbf{S_A}$ in these five synthetic datasets are $0.5k$, $1k$, $2k$, $10k$, $20k$, respectively. Note that there are only two types of events we simulated in synthetic datasets. In practice, a real dataset typically includes more than hundreds of events types. Therefore, we believe $20k$ events of two types is enough to represent the real application scenarios in miniature.

Synthetic Data Evaluation. Since all of our proposed algorithms are based on the EM algorithm, which cannot guarantee the global optimum [5], we define a batch operation to avoid this problem as much as possible. Specifically, every 10 rounds execution of the algorithm with different initial parameters chosen at random is regarded as a batch. For each batch, we choose the output with the maximum likelihood among 10 rounds as the result of a batch. For each synthetic dataset, we conduct five such batches on it, and calculated the average values of the results of five batches as the final result. Table 2 shows the outcome of experiments running *EMLag*, *winEMLag*, and *appEMLag* on such five synthetic datasets with different parameters settings, respectively.

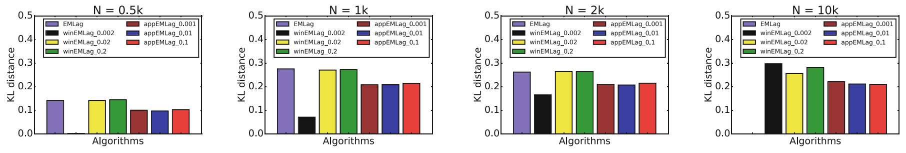
Each algorithm terminates execution when it satisfies one of the following conditions: (1) it converges; (2) the number of iterations exceeds 500; or (3) the differences of all learned parameters between two adjacent iterations are less than 10^{-5}. *winEMLag* algorithm takes one more parameter λ as its input, where λ determines the length of windows used in the algorithm. In our experiments, we set λ to 0.002, 0.02 and 0.2. Similarly, *appEMLag* algorithm has a predefined parameter ϵ, which is used to calculate the proportion of the neglected part during the parameter estimation of each iteration. In order to evaluate the performance of *appEMLag* algorithm sufficiently, ϵ is set to 0.001, 0.01, and 0.1. As shown in Table 2, we find that parameters μs learned by *EMLag*, *winEMLag*, and *appEMLag* are quite close to the ground truth.

In addition, for evaluating the difference between the distributions of time lags given by the ground truth and learned by our proposed algorithms shown in Table 2, we introduce Kullback-Leibler (KL) divergence [14]. Figure 2 shows the evaluation results with various sizes of dataset from 0.5k to 10k, respectively. Here we can see, all proposed algorithms could effectively discovery the distributions of time lags from event sequences. Moreover, compared with *winEMLag*, *appEMLag* algorithm performs better in terms of KL divergence.

Table 2. The experimental results for various synthetic datasets with sizes from $0.5k$ to $20k$. The values of μ and σ^2 for "ground truth" are given in advance; $\overline{\mu}$ and $\overline{\sigma}^2$ represent the average values of μ and σ^2; LL_{opt} is the maximum log-likelihood obtained by running the algorithm; Entries with "N/A" are not available since they take more than 7 days to get corresponding results.

Dataset	Ground Truth			EMLag			winEMLag $\lambda = 0.002$			winEMLag $\lambda = 0.02$		
	(μ_1, σ_1^2)	(μ_2, σ_2^2)	LL_{opt}	$(\overline{\mu}_1, \overline{\sigma}_1^2)$	$(\overline{\mu}_2, \overline{\sigma}_2^2)$	LL_{opt}	$(\overline{\mu}_1, \overline{\sigma}_1^2)$	$(\overline{\mu}_2, \overline{\sigma}_2^2)$	LL_{opt}	$(\overline{\mu}_1, \overline{\sigma}_1^2)$	$(\overline{\mu}_2, \overline{\sigma}_2^2)$	LL_{opt}
$N = 0.5k$	(28.41, 17.97)	(59.16, 18.94)	-	(28.14, 10.02)	(58.49, 14.65)	−4222.01	(28.52, 19.06)	(59.04, 18.48)	−1442.53	(28.11, 10.04)	(58.50, 14.59)	−2267.04
$N = 1k$	(35.67, 24.89)	(65.93, 33.12)	-	(35.72, 10.55)	(66.32, 25.03)	−9188.64	(35.16, 22.31)	(64.60, 24.45)	−3402.28	(35.71, 10.62)	(66.24, 24.91)	−5275.52
$N = 2k$	(35.68, 14.34)	(73.70, 28.30)	-	(35.59, 6.97)	(72.68, 17.75)	−19353.02	(35.82, 11.61)	(72.47, 15.79)	−7367.57	(35.58, 6.94)	(72.59, 18.06)	−11526.52
$N = 10k$	(46.83, 34.41)	(69.96, 29.62)	-	N/A	N/A	N/A	(47.20, 24.30)	(70.13, 12.40)	−53044.11	(47.03, 22.17)	(69.99, 13.64)	−76051.98
$N = 20k$	(55.46, 32.51)	(75.66, 29.42)	-	N/A	N/A	N/A	(55.62, 14.58)	(75.19, 15.79)	−117563.57	(55.39, 13.01)	(75.15, 16.91)	−163582.36

Dataset	winEMLag $\lambda = 0.2$			appEMLag $\epsilon = 0.001$			appEMLag $\epsilon = 0.01$			appEMLag $\epsilon = 0.1$		
	$(\overline{\mu}_1, \overline{\sigma}_1^2)$	$(\overline{\mu}_2, \overline{\sigma}_2^2)$	LL_{opt}	$(\overline{\mu}_1, \overline{\sigma}_1^2)$	$(\overline{\mu}_2, \overline{\sigma}_2^2)$	LL_{opt}	$(\overline{\mu}_1, \overline{\sigma}_1^2)$	$(\overline{\mu}_2, \overline{\sigma}_2^2)$	LL_{opt}	$(\overline{\mu}_1, \overline{\sigma}_1^2)$	$(\overline{\mu}_2, \overline{\sigma}_2^2)$	LL_{opt}
$N = 0.5k$	(28.12, 10.03)	(58.44, 14.64)	−3408.06	(28.98, 11.51)	(59.31, 13.84)	−1581.37	(28.85, 11.56)	(59.27, 13.56)	−1573.18	(28.83, 11.57)	(59.03, 13.14)	−1561.07
$N = 1k$	(35.70, 10.59)	(66.25, 24.92)	−7561.03	(36.15, 13.51)	(67.03, 21.41)	−3318.73	(36.06, 13.21)	(66.93, 21.53)	−3310.18	(36.05, 12.99)	(66.82, 21.25)	−3290.54
$N = 2k$	(35.55, 6.92)	(72.45, 18.77)	−16102.33	(36.11, 9.52)	(74.27, 14.52)	−6229.23	(36.04, 9.59)	(74.25, 14.45)	−6211.82	(35.92, 9.45)	(74.01, 14.02)	−6183.27
$N = 10k$	(47.08, 23.21)	(70.09, 12.87)	−98914.56	(47.56, 19.82)	(71.03, 17.51)	−33681.02	(47.46, 19.42)	(70.87, 17.78)	−33585.87	(47.42, 19.43)	(70.69, 17.38)	−33416.45
$N = 20k$	N/A	N/A	N/A	(56.97, 17.23)	(76.55, 14.40)	−65431.36	(56.59, 16.22)	(76.35, 14.84)	−65266.24	(56.67, 16.43)	(76.35, 14.57)	−64848.79



(a) Size = 0.5k (b) Size = 1k (c) Size = 2k (d) Size = 10k

Fig. 2. The KL distance between the ground truth and the one learned by each algorithm over different datasets.

The efficiency comparison between all proposed algorithms is measured by the CPU running time. As shown in Fig. 3, the time cost of *winEMLag* and *appEMLag* are much less than the *EMLag* algorithm. Since both parameter ϵ and λ could effectively decrease the number of events needed to be considered in each iteration, the efficiency of these two algorithms are satisfactory.

In summary, based on the extensively comparative experiments on synthetic data, all of our proposed algorithms have the capabilities for finding time lags from fluctuating events effectively. Two approximate algorithms *winEMLag* and *appEMLag* could achieve a good balance in terms of accuracy and efficiency.

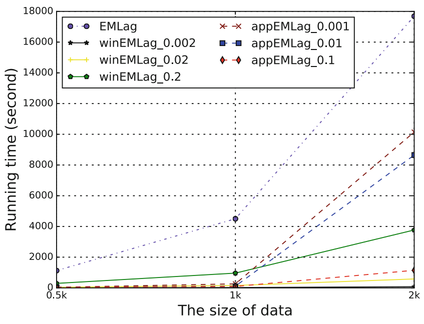


Fig. 3. Time cost comparison

6.2 Real Data

We employ two real datasets collected from several IT outsourcing centers by IBM Tivoli monitoring system [12] to verify the performance of our proposed algorithms in real application scenarios. Each dataset is a collection of system events generated by the automatic monitoring system running on servers. Most of these events are system alerts triggered by some monitoring situations, for example, the disk capacity is almost full. Table 3 lists the statistical information of these two real event datasets.

Table 3. Real event datasets

Dataset	Time span	# of events	# of types
Dataset1	32 days	$100k$	104
Dataset2	54 days	$1,000k$	136

In order to discover the time lags of temporal dependencies more effectively and efficiently, we choose *appEMLag* algorithm with parameter setting $\epsilon = 0.001$ to deal with these two real event datasets. For increasing the probability of acquiring the global optimal value, we run *appEMLag* in a batch of 30 rounds with randomly initialize parameters every round. Table 4 provides a snippet of some discovered temporal dependencies with multiple time lags from two real datasets. We employ the metric signal-to-noise ratio [24], a concept in signal processing domain, to evaluate the impact of noise relative to the expected time lag. Signal-to-noise ratio (SNR) can be calculated as the ratio of the expect time lag μ to the standard deviation σ. Here, we use the average value of SNR for two discovered distributions as the measure.

The time lags of temporal dependency $AIX\_HW\_Error \rightarrow_L NV390MSG\_MVS$ discovered from dataset1 follow two types of normal distribution, one is $\mu = 55.41$ and $\sigma^2 = 0.39$, and the other one is $\mu = 93.09$ and $\sigma^2 = 0.32$. Compared with algorithms proposed in [30], which can only find one time lag

Table 4. Snippet of discovered temporal dependencies with multiple time lags

Dataset	Temporal dependency	(μ_1, σ_1^2)	(μ_2, σ_2^2)	Ave. SNR
Dataset1	*AIX_HW_Error* $\rightarrow_L$ *NV390MSG_MVS*	(55.41, 0.39)	(93.09, 0.32)	126.64
	generic_postemsg $\rightarrow_L$ *NV390MSG_AO_Platform_Server*	(63.55, 9.34)	(18.23, 16.37)	12.65
	generic_postemsg $\rightarrow_L$ *Sentry2_0_diskusedpct*	(219.38, 10.18)	(146.27, 10.41)	57.05
	NV390MSG_AO_Platform_Server $\rightarrow_L$ *Info_set_ticket_number_using_eventid*	(2.62, 4.49)	(18.09, 2981.51)	0.78
	MQ_CONN_NOT_AUTHORIZED $\rightarrow_L$ *ITM_NT_Services*	(1912.78, 47.63)	(88.16, 34.00)	146.14
	Ticket_Retry $\rightarrow_L$ *TEC_Notice*	(360.59, 41.20)	(269.17, 44.76)	48.21
	TEC_Error $\rightarrow_L$ *ITM_KGB_AVAILABILITY*	(468.66, 76.77)	(434.34, 36.88)	62.50
Dataset2	*Generic_Source_Event* $\rightarrow_L$ *Candle_Universal_Messages*	(383.49, 104.02)	(43.17, 24.80)	23.13
	ITM_Process $\rightarrow_L$ *PATROL_APP*	(461.84, 0.93)	(168.13, 2433.03)	241.16
	ITM_Process $\rightarrow_L$ *OV_IF_Down*	(36.66, 256.39)	(244.63, 261.73)	8.71

with the expected time lag $\mu = 33.89$ and the variance $\sigma^2 = 1.95$, our method discover one more type of time lag. Moreover, since the variances of these two discovered time lags are quit small, these two time lags are very close to the true time lags.

The temporal dependency *generic_postemsg* $\rightarrow_L$ *Sentry2_0_diskusedpct* has two time lags with different expected time lags μ and similar variances σ^2. Event *Sentry2_0_diskusedpct* appears 2.5 or 3.5 min later after *generic_postemsg* occurs. Conversely, the expected time lags between *ITM_KGB_AVAILABILITY* and *TEC_Error* are similar, while the variances are different. Because the expected time lags are very similar with each other, it is not trivial to capture two normal distributions from large datasets. Previous temporal dependency mining methods only return one constant time lag as the result, due to they ignore the existence of the noise and are not able to distinguish two very similar time lags.

The variances of the time lags between *Info_set_ticket_number_using_eventid* and *NV390MSG_AO_Platform_Server* in dataset1 are quite large relative to the expected time lags, since the average SNR is less than 1. For every single time lag, the variance is still relative large. Hence, we think this is a week dependency between these two events due to the discovered time lags contain too much noise.

We find the time lags of temporal dependency *ITM_Process* $\rightarrow_L$ *PATROL_APP* in dataset2 are quite different with other dependencies. Specifically, one time lag has large value of the expect time lag μ and small value of the variance σ^2, and the other one has small value of μ and large value of σ^2. Both of them are very difficult for previous inter-arrival pattern mining methods to discover, where the inter-arrival time lags are small time lags. In our methods, we use the expected time lag μ and its variance σ^2 to find multiple interleaved time lags.

7 Conclusions

In this paper, we study the problem of discovering multiple time lags of temporal dependencies over event sequences, where the time lags between two pairwise events are fluctuating since the existence of the noise. To solve this problem, an EM-based algorithm is proposed to capture the distribution of time lags. We also propose two approximate algorithms for speeding up the time lag mining process. Extensive empirical studies on both synthetic and real datasets demonstrate the efficiency and effectiveness of our proposed algorithms.

In future work, we plan to implement distributed versions of our algorithms for handling applications with massive data. Furthermore, mining dependencies among multiple events other than pairwise events is also attractive to us.

References

1. Aggarwal, C.C.: Data Streams: Models and Algorithms, vol. 31. Springer, Heidelberg (2007). https://doi.org/10.1007/978-0-387-47534-9
2. Ailamaki, A., Faloutos, C., Fischbeck, P.S., Small, M.J., VanBriesen, J.: An environmental sensor network to determine drinking water quality and security. ACM SIGMOD Rec. **32**(4), 47–52 (2003)
3. Arnold, A., Liu, Y., Abe, N.: Temporal causal modeling with graphical granger methods. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 66–75. ACM (2007)
4. Bahadori, M.T., Liu, Y.: An examination of practical granger causality inference. In: Proceedings of the 2013 SIAM International Conference on Data Mining, pp. 467–475. SIAM (2013)
5. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Heidelberg (2006)
6. Bouandas, K., Osmani, A.: Mining association rules in temporal sequences. In: IEEE Symposium on Computational Intelligence and Data Mining CIDM 2007, pp. 610–615. IEEE (2007)
7. Dhurandhar, A.: Learning maximum lag for grouped graphical granger models. In: 2010 IEEE International Conference on Data Mining Workshops (ICDMW), pp. 217–224. IEEE (2010)
8. Du, X., Jin, R., Ding, L., Lee, V.E., Thornton Jr, J.H.: Migration motif: a spatial-temporal pattern mining approach for financial markets. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1135–1144. ACM (2009)
9. Golab, L., Karloff, H., Korn, F., Saha, A., Srivastava, D.: Sequential dependencies. Proc. VLDB Endow. **2**(1), 574–585 (2009)
10. Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., Hsu, M.C.: FreeSpan: frequent pattern-projected sequential pattern mining. In: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 355–359. ACM (2000)
11. Hellerstein, J.L., Ma, S., Perng, C.S.: Discovering actionable patterns in event data. IBM Syst. J. **41**(3), 475–493 (2002)
12. IBM Tivoli Monitoring. http://www-01.ibm.com/software/tivoli/
13. Jansen, R., Yu, H., Greenbaum, D., Kluger, Y., Krogan, N.J., Chung, S., Emili, A., Snyder, M., Greenblatt, J.F., Gerstein, M.: A bayesian networks approach for predicting protein-protein interactions from genomic data. Science **302**(5644), 449–453 (2003)
14. Kullback, S., Leibler, R.A.: On information and sufficiency. Ann. Math. Stat. **22**(1), 79–86 (1951)

15. Laxman, S., Sastry, P.S.: A survey of temporal data mining. Sadhana **31**(2), 173–198 (2006)
16. Li, T.: Event Mining. Chapman and Hall/CRC, Boca Raton (2015)
17. Li, T., Liang, F., Ma, S., Peng, W.: An integrated framework on mining logs files for computing system management. In: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, pp. 776–781. ACM (2005)
18. Li, T., Ma, S.: Mining temporal patterns without predefined time windows. In: Fourth IEEE International Conference on Data Mining ICDM 2004, pp. 451–454. IEEE (2004)
19. Li, T., Peng, W., Perng, C., Ma, S., Wang, H.: An integrated data-driven framework for computing system management. IEEE Trans. Syst. Man Cybern.-Part A: Syst. Hum. **40**(1), 90–99 (2010)
20. Lozano, A.C., Abe, N., Liu, Y., Rosset, S.: Grouped graphical granger modeling methods for temporal causal modeling. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 577–586. ACM (2009)
21. Ma, S., Hellerstein, J.L.: Mining partially periodic event patterns with unknown periods. In: Proceedings 17th International Conference on Data Engineering 2001, pp. 205–214. IEEE (2001)
22. Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.: Prefixspan: mining sequential patterns by prefix-projected growth. In: Proceedings of the 17th International Conference on Data Engineering, pp. 215–224. IEEE Computer Society, Washington, DC, USA (2001)
23. Peng, W., Perng, C., Li, T., Wang, H.: Event summarization for system management. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1028–1032. ACM (2007)
24. Schroeder, D.J.: Astronomical Optics. Academic press, Cambridge (1999)
25. Song, L., Kolar, M., Xing, E.P.: Time-varying dynamic Bayesian networks. In: Advances in Neural Information Processing Systems, pp. 1732–1740 (2009)
26. Srikant, R., Agrawal, R.: Mining sequential patterns: generalizations and performance improvements. In: Apers, P., Bouzeghoub, M., Gardarin, G. (eds.) EDBT 1996. LNCS, vol. 1057, pp. 1–17. Springer, Heidelberg (1996). https://doi.org/10.1007/BFb0014140
27. Tan, P.N., et al.: Introduction to Data Mining. Pearson Education India, London (2006)
28. Tang, L., Li, T., Shwartz, L.: Discovering lag intervals for temporal dependencies. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 633–641. ACM (2012)
29. Zeng, C., Wang, Q., Wang, W., Li, T., Shwartz, L.: Online inference for time-varying temporal dependency discovery from time series. In: 2016 IEEE International Conference on Big Data (Big Data), pp. 1281–1290 (2016)
30. Zeng, C., Tang, L., Li, T., Shwartz, L., Grabarnik, G.Y.: Mining temporal lag from fluctuating events for correlation and root cause analysis. In: 2014 10th International Conference on Network and Service Management (CNSM), pp. 19–27. IEEE (2014)