12th IFAC/IEEE Workshop on Discrete Event Systems
Cachan, France. May 14-16, 2014

Pattern Mining for Predicting Critical Events from Sequential Event Data Log

Jun Chen [*] and **Ratnesh Kumar** [*]

[*] *Department of Electrical and Computer Engineering*
Iowa State University, Ames, IA 50011 USA
e-mail: {junchen,rkumar}@iastate.edu.

Abstract: This paper studies the mining of patterns for predicting critical events from observed ordered event data, where the observations can contain interleaving from non-predictor and other predictor event sequences. These are characteristics of many practical applications such as monitoring in power systems or telecommunication networks, as well as computational biology. For settings where system behaviors are affected by noise, a critical event can sometimes occur without its predictor executed prior to it, and we propose algorithm to recursively compute the frequency that a predictor candidate precedes the critical event. This we use for identifying a predictor, and study the performance of such a scheme. We also consider the noise-free settings, in which a critical event occurs only after the execution of its predictor, and propose an algorithm to recursively compute the set of maximal predictors for each critical event.

Keywords: Pattern mining, finite automata, alarm monitoring, common subsequence, algorithm.

1. INTRODUCTION

For certain monitoring applications such as failure diagnosis (Chen and Kumar (2013a,b,c)) and failure prediction (Kumar and Takai (2010); Chen and Kumar (2014b)), knowledge of system model is essential for any detection algorithm. However, in many scenarios, such as business processes, offline determination of a detailed model describing a certain process is challenging and many times impossible since the system may be overly complex, e.g., vast legacy software. The task is further complicated by the fact that, there exists discrepancy between a *prescriptive* model describing how process is expected to work and a *driving* model according to which the system evolves. Thus even the prescriptive models may not be used (while driving models are overly complex to obtain). Therefore, instead of performing offline process/workflow modeling, there have been efforts towards online process/workflow mining, where the idea is to develop a model describing the underlying process, given the workflow logs in terms of ordered event data. See the next section for literature information on workflow/process mining.

The task of mining workflow/system model is computationally hard (Hsu et al. (2012)), and instead, the simpler problem of pattern mining intends to find certain sequence patterns that precede certain critical events. Such predictor pattern identification problem has many real-world applications such as alarm log processing in power systems and telecommunication networks, and genetic motif discovery in computational biology (e.g., Liu et al. (1995), Chudova and Smyth (2002) and Bailey and Elkan (1995)).

[⋆] The research was supported in part by the National Science Foundation under the grants, NSF-ECCS-0801763, NSF-ECCS-0926029, and NSF-CCF-1331390.

The authors Agrawal and Srikant (1995); Pei et al. (2004); Han et al. (1999) and Cheng et al. (2005) study a type of pattern mining problem where a pattern is simply a repetitive sequence of events, not necessarily associated with a critical event.

In this paper, we study the mining of patterns for predicting critical events from observed ordered event data, where the observations can contain interleaving from non-predictor and other predictor event sequences. Two different scenarios, noisy and noise-free, are considered. In the settings where system behaviors are affected by noise, there is a non-zero probability that a critical event can occur without being preceded by its predictor. We propose an algorithm to recursively compute the frequency that a predictor candidate precedes the critical event. A candidate is deemed to be a predictor for certain critical event if its frequency of occurrence preceding to that critical event is above a user-specified confidence level and it is "maximal", i.e., none of its *supersequence* is a predictor candidate.

We also consider the noise-free settings, in which a critical event can occur only after the execution of its predictor, and the predictor mining reduces to finding the maximal common subsequences, which has been studied by Hunt and Szymanski (1977); Hirschberg (1975); Allison and Dix (1986). We propose another recursive algorithm to compute the set of maximal predictors for each critical event. The related work by Wang and Johnson (2007) considers a similar problem of pattern mining from event sequence log, where the patterns are assumed to be contiguous event sequences of the system model. In this paper, we relax such assumption about contiguity by allowing interleaving among the predictors as well as the non-predictor sequences.

The contributions of this paper are summarized as follows:

- Compared to prior works, we allow the observed ordered event data to contain interleaving from non-predictor and other predictor event sequences;
- We propose recursive algorithms for mining predictor patterns for both noisy and noise-free settings;
- For noisy setting, we provide the existence of a bound for the length of event log that guarantees the proposed algorithm to output correct predictor within a desired error probability.

The rest of this paper is organized as follows: Section 2 briefly describes some related work on process mining. The notation and some preliminaries are presented in Section 3. The formulation of the pattern mining problem is presented in Section 4, which also provides the recursive algorithm for the computing the predictors in setting of noisy observations. Section 5 considers the noise-free observations and provides an algorithm to recursively compute the set of maximal predictors for each critical event. The paper is concluded in Section 6 with directions for future work.

2. RELATED WORK ON PROCESS MINING

While pattern mining attempts to discover the signatures for critical events, process mining or model identification (Cabasino et al. (2011)) is a more ambitious endeavor that tries to uncover the entire process model. Being again driven by the observed ordered event data, it is a related topic of research, and here we provide a short summary for interested readers. An algorithm to extract a process model in form of a Petri net from event sequence log was introduced in van der Aalst et al. (2003, 2004), where the set of underlying model that can be extracted from event sequence log is also given. Cook and Wolf (1998) investigated the process mining problem in the context of software engineering processes, under the framework of *grammar inference* by Gold (1967, 1978) and Angluin (1987). Cook and Wolf (1998) presented several approaches: i) *RNet*, which is based on neural network; ii) *Ktail*, which outputs a finite state automaton whose state space is given by the set of equivalence classes of traces that have the same k-step extensions; and iii) *Markov* method, which assumes that the underlying model is Markovian with order at most 2, whose dependencies are statistically deduced when the occurrence frequencies of contiguity for event pairs is above a user-specified threshold.

As indicated by Cook and Wolf (1998), the process mining can be cast into a grammar inference problem as in Gold (1967, 1978) and Angluin (1987). An algorithm for grammar inference, called L^* algorithm, was proposed by Angluin (1987) which requires a membership oracle as well as an equivalence oracle, the former of which identifies the membership of a given trace while the latter can confirm the correctness of a postulated grammar and return a counterexample if the postulate is false. The stochastic grammar identification was addressed in Carrasco and Oncina (1994) by merging the equivalent nodes in a complete prefix tree, where two nodes are deemed equivalent if they have equivalent successors and the distance between their distributions over the set of successors is within a tolerance. The identification of stochastic grammar can also be addressed in the framework of source identification. Cybenko and Crespi (2011) consider the identification of a hidden Markov model from the observed symbols sequence using *nonnegative matrix factorization*. The proposed algorithm in Cybenko and Crespi (2011) computes a frequency matrix by computing, for each pair of sequence of observed symbols, the occurrence frequency that the first sequence of the pair is followed immediately by the second sequence of that pair. The order of the hidden Markov model as well as the state transition matrix are then estimated from this frequency matrix, by computing its *positive rank* and nonnegative matrix factorization, which in general is approximated by optimizing an objective function consisting of a type of divergence. It turns out that (see Hsu et al. (2012)) identification of hidden Markov models from data is computationally hard. Hence the literature also explores the simpler problem of order estimation of a Markov model. Merhav et al. (1989) studied the estimation of the order of a fully observable Markov process, whereas Liu and Narayan (1994) investigated the order estimation of hidden Markov model. The order estimation problem examines the dependency of data in the observed sequence with its preceding history, and is relevant for data compression (source coding) for storage and communication.

3. NOTATIONS AND PRELIMINARIES

For an event set Σ, define $\overline{\Sigma} := \Sigma \cup \{\epsilon\}$, where ϵ denotes "no-event". The set of all finite length event sequences over Σ, including ϵ, is denoted as Σ^*. A *trace* is a member of Σ^*, i.e., a trace is a sequence of events, and a *language* is a subset of Σ^*. We use $s \leq t$ to denote that $s \in \Sigma^*$ is a prefix of $t \in \Sigma^*$, $pr(s)$ to denote the set of all prefixes of s, and $|s|$ to denote the length of s or the number of events in s. For $\sigma \in \Sigma$ and $s \in \Sigma^*$, we use $\sigma \in s$ to denote that the event σ is an element of the trace s. For $\sim \in \{<, \leq, >, \geq, =\}$ and $n \in \mathbb{N}$, where $\mathbb{N}$ denotes the set of all nonnegative integers, define $\Sigma^{\sim n} := \{s \in \Sigma^* \mid |s| \sim n\}$ and denote $\Sigma^{=n}$ as Σ^n for simplicity. For any $s = \sigma_1 \dots \sigma_{|s|} \in \Sigma^*$, denote as $s^{-1} := \sigma_{|s|} \dots \sigma_1$ for the *reverse* of s. $t \in \Sigma^{\leq |s|}$ is said to be a *subsequence* of s, denoted $t \ll s$, if there exists indices $1 \leq i_1 \leq \dots \leq i_{|t|} \leq |s|$ such that $t = \sigma_{i_1} \dots \sigma_{i_{|t|}}$, and in this case we also call s as a *supersequence* of t. t is said to be a common subsequence of s_1 and s_2 if $t \ll s_1$ and $t \ll s_2$. The interleaving product of a pair of traces $s, t \in \Sigma^*$, denoted $s \bowtie t$, is defined recursively as follows: $\epsilon \bowtie \epsilon := \epsilon$; $\forall s, s' \in \Sigma^*, \sigma, \sigma' \in \Sigma : s\sigma \bowtie s'\sigma' := (s\sigma \bowtie s').\sigma' + (s \bowtie s'\sigma')\sigma$. The interleaving product of two languages L_1 and L_2 is given by, $L_1 \bowtie L_2 := \{s \bowtie t \mid s \in L_1, t \in L_2\}$.

A *finite state automaton* is a tuple $G = (X, \Sigma, \alpha, X_0)$, where X is the set of states, Σ is the set of events, $\alpha : X \times \overline{\Sigma} \to 2^X$ is the transition function, and $X_0 \subseteq X$ is the set of initial states. G is deterministic if $|X_0| = 1$, i.e., a unique initial state, and $\forall x \in X, \sigma \in \Sigma, |\alpha(x, \sigma)| \leq 1$ and $|\alpha(x, \epsilon)| = 0$, i.e., each state has at most one transition on each event and no transition on "no-event"; otherwise G is called nondeterministic. A path of G is a sequence of transitions $(x_1, \sigma_1, x_2, \dots, \sigma_{n-1}, x_n)$ such that $\sigma_i \in \overline{\Sigma}$ and $x_{i+1} \in \alpha(x_i, \sigma_i)$ for each $1 \leq i \leq n-1$. A path is called a cycle if $x_n = x_1$. For any $x \in X$, define the ϵ-closure of x, denoted as $\epsilon^*_G(x)$, as the set of all states that can be reached by zero or more ϵ transitions from state x, which can be recursively defined as:

$$x \in \epsilon^*_G(x); \quad x' \in \epsilon^*_G(x) \Rightarrow \alpha(x', \epsilon) \subseteq \epsilon^*_G(x).$$

Then the transition function α can be generalized to $\alpha : X \times \Sigma^* \to 2^X$ recursively as follows: $\forall x \in X, s \in \Sigma^*, \sigma \in \Sigma$,

$$\alpha(x, \epsilon) = \epsilon^*_G(x); \quad \alpha(x, s\sigma) = \epsilon^*_G(\alpha(\alpha(x, s), \sigma)).$$

We define the language generated by G as $L(G) := \{s \in \Sigma^* \mid \exists x_0 \in X_0, \alpha(x_0, s) \neq \emptyset\}$. Given an automaton $G = (X, \Sigma, \alpha, X_0)$, $G' = (X', \Sigma', \alpha', X'_0)$ is said to be a subautomaton of G if $X' \subseteq X$, $\Sigma' \subseteq \Sigma, X'_0 \subseteq X_0$ and $\forall x \in X', \sigma \in \Sigma', \alpha'(x, \sigma) \subseteq \alpha(x, \sigma)$.

4. PROBLEM FORMULATION AND APPROACH

Suppose the process generating the observed event sequence is driven by a finite state automaton $G = (X, \Sigma, \alpha, X_0)$, but this model is unknown, and so the predictor-patterns must be learned by monitoring the system evolution. A subset of event set $\Sigma_\theta \subseteq \Sigma$ is designated as critical events that indicate certain key events of interest (e.g., occurrence of fault, structure variation of DNA sequence). For each critical event $\theta \in \Sigma_\theta$, define an ordered sequence of events $\pi_\theta \in \Sigma^*$ as a predictor-pattern (or simply predictor) for θ, if all traces in $L(G)$, ending in critical event θ, contain π_θ as a subsequence. Note that the notion of predictor can be useful for the purpose of fault detection (see for example Jiang and Kumar (2004, 2006); Chen and Kumar (2014a)) and fault prediction (see for example Genc and Lafortune (2009), Kumar and Takai (2010) and Chen and Kumar (2014b)). The formal definition of predictor is given as:

Definition 1. An ordered event sequence π_θ is a *predictor* for $\theta \in \Sigma_\theta$, if $\forall u\theta s\theta \in L(G)$ such that $\theta \notin s$, it holds that π_θ is a subsequence of s, namely, $\pi_\theta \ll s$. If the underlying model is subject to probabilistic distribution, then given $\mu \leq 1$, an ordered event sequence π_θ is a predictor if $\forall u\theta s\theta \in L(G)$ such that $\theta \notin s$, it holds that $Pr(\pi_\theta \ll s) \geq \mu$. Moreover, π_θ is a *maximal* predictor for $\theta \in \Sigma_\theta$ if none of its supersequence is a predictor for θ.

Note that, unlike Wang and Johnson (2007), the definition of predictor requires it to be a subsequence, instead of a subtrace (i.e., a contiguous subsequence), of the language $L(G)$. If we use a "cyclical" automaton G_θ with a single cycle over the trace $\pi_\theta\theta$, i.e., $L(G_\theta) = pr((\pi_\theta\theta)^*)$, then $L(G)$ can be seen as a subset of the interleaving product of $\{L(G_\theta), \theta \in \Sigma_\theta\}$ and $(\Sigma - \Sigma_\theta)^*$. Note that G_θ is not necessarily a subautomaton of G. Also, the example illustrates that while the system is executing a trace, events from predictor versus non-predictor subsequences can interleave.

Example 1. An example of automaton G is given in Fig. 1, with $\Sigma = \{a, b, c, e_1, e_2\}$ and $\Sigma_\theta = \{e_1, e_2\}$. The maximal predictor for e_1 is $\pi_{e_1} = ac$ and the maximal predictor for e_2 is $\pi_{e_2} = bcb$. There two predictors can be represented by the cyclical automata in Fig. 2. One example of the observed event log is $\ell = ace_1cbbcbe_2cace_1bbbcbe_2aabace_1aace_1be_2$, where π_{e_1} and π_{e_2} and non-predictor events are all interleaved. ■

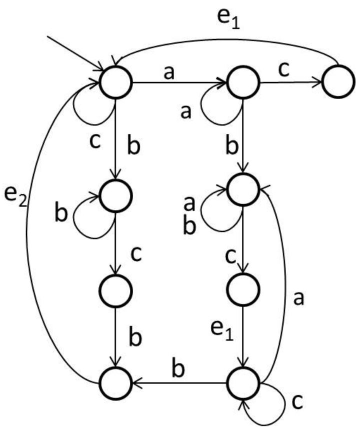

Fig. 1. System G.

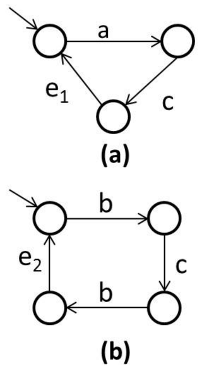

Fig. 2. Cyclical automata: (a) G_{e_1}; (b) G_{e_2}.

If some critical events can only occur for a finite number of times, then the system may not generate sufficient data for mining its predictor, and so we assume G is such that every critical event can be repeated infinitely often and has at least one predictor. In fact, our proposed algorithm is capable of mining only the "ergodic" behavior of the underlying process. Then the objective is to identify π_θ for each $\theta \in \Sigma_\theta$ given an event trace or log ℓ generated by G (whose model as we mentioned above is unknown).

4.1 Approach for noisy pattern mining

Given an observed event sequence log ℓ, to mine the dependency between critical event $\theta \in \Sigma_\theta$ and certain subsequence (as predictor candidate), we first remove all critical events in $\Sigma_\theta - \theta$ from ℓ, and denote the resulting event sequence as ℓ_θ. Let n_θ be the number of occurrence of θ in ℓ_θ. Then there exists traces $s^\theta_1, s^\theta_2, \ldots, s^\theta_{n_\theta}, s^\theta_{n_\theta+1} \in (\Sigma - \Sigma_\theta)^*$ such that $s^\theta_1\theta s^\theta_2\theta \ldots s^\theta_{n_\theta}\theta s^\theta_{n_\theta+1} = \ell_\theta$, i.e., s^θ_k is the longest subsequence of non-critical events of ℓ_θ between the $(k-1)$th and kth critical event θ. Denote $m := \max_{\theta\in\Sigma_\theta, k=1,\ldots,n_\theta} |s^\theta_k|$. For each $\theta \in \Sigma_\theta$ and $s \in (\Sigma - \Sigma_\theta)^{\leq m}$, compute

$$f_\ell(s, \theta) = \frac{|\{s^\theta_k \mid s \ll s^\theta_k\}|}{n_\theta}, \tag{1}$$

which is the fraction that a subsequence $s \in (\Sigma - \Sigma_\theta)^{\leq m}$ appears prior to θ in the observed event sequence ℓ_θ. Then we propose the following as an estimator for the set of maximal predictors for $\theta \in \Sigma_\theta$:

$$\hat{\pi}_\theta := \{s \in (\Sigma - \Sigma_\theta)^{\leq m} \mid f_\ell(s,\theta) \geq \nu,$$
$$\nexists t \neq s \text{ such that } (f_\ell(t,\theta) \geq \nu) \wedge (s \ll t)\}, \quad (2)$$

where ν is a confidence level chosen to be smaller than the one used in Definition 1 (see the proof for Theorem 1).

According to Definition 1, a given critical event can have multiple maximal subsequences as predictors. Therefore $\hat{\pi}_\theta$ as defined in (2) is a set, and not necessarily a unique element.

The following algorithm recursively computes $f_\ell(s,\theta)$ for each $\theta \in \Sigma_\theta$ and $s \in (\Sigma - \Sigma_\theta)^{\leq m}$. The algorithm simply updates the above frequency at the arrival of each critical event, and otherwise simply tracks the trace that the system executes, starting from the last critical event until a new critical event. There are three inputs to the algorithm, namely, the event set Σ, the set of critical event $\Sigma_\theta \subseteq \Sigma$ and an event log ℓ, and one output, namely, estimator $\hat{\pi}_\theta$ as the set of maximal predictors for each $\theta \in \Sigma_\theta$.

Algorithm 1. For given event set Σ and critical event sets $\Sigma_\theta \subseteq \Sigma$:

```
1 Initialization:
    • $\forall \theta \in \Sigma_\theta$: $s_\theta := \epsilon, n_\theta := 0, m_\theta := 1$.
2 Upon arrival of new $\sigma \in \Sigma - \Sigma_\theta$, do:
    • $\forall \theta \in \Sigma_\theta$: $s_\theta := s_\theta \sigma$.
3 Upon arrival of new $\theta \in \Sigma_\theta$, do:
    • $m_\theta := \max\{m_\theta, |s_\theta|\}, \forall s \in (\Sigma - \Sigma_\theta)^{\leq m_\theta}$:
      $$f_\ell(s,\theta) := \begin{cases} \dfrac{n_\theta f_\ell(s,\theta)+1}{n_\theta+1} & \text{if } s \ll s_\theta \\ \dfrac{n_\theta f_\ell(s,\theta)}{n_\theta+1} & \text{otherwise,} \end{cases}$$
      where on the right side $f_\ell(s,\theta)$ is taken to be zero
      if it is undefined.
    • $s_\theta := \epsilon, n_\theta := n_\theta + 1$.
4 At any point of time, the estimator of maximal predictor
  is given by (2).
```

Remark 1. For each newly arrived non-critical event $\sigma \in \Sigma - \Sigma_\theta$, the complexity for updating s_θ is $O(|\Sigma_\theta|)$ and for an event log ℓ the overall complexity for this step is $O(|\ell| \times |\Sigma_\theta|)$. For each newly arrived critical event $\theta \in \Sigma_\theta$, it has $O(|\Sigma - \Sigma_\theta|^m)$ predictor candidates, and needs $O(m^2|\Sigma - \Sigma_\theta|^m)$ to update $f_\ell(s,\theta)$ for each predictor candidate s (complexity for checking whether $s \ll s_\theta$ is $O(|s| \times |s_\theta|) \leq O(m^2)$). Therefore for an event log ℓ the overall complexity for this step is $O(n \times |\Sigma_\theta| \times m^2|\Sigma - \Sigma_\theta|^m)$, where $n := \max_\theta n_\theta$. Since $n \leq |\ell|$, it follows that the complexity for processing an event log ℓ is given by $O(|\ell| \times |\Sigma_\theta| \times m^2|\Sigma - \Sigma_\theta|^m)$, where the parameter m is upper bounded by the number of events G can execute between a pair of same critical events.

To evaluate the performance of Algorithm 1, we assume that for a particular $\theta \in \Sigma_\theta$, a true maximal predictor exists, denoted as π_θ, i.e., the probability of π_θ occurring prior to θ exceeds μ.

Theorem 1. Assume the underlying model G is ergodic. Then for any critical event $\theta \in \Sigma_\theta$ and positive τ, there exists $n \in \mathbb{N}$, such that for an event log ℓ longer than n, the probability of outputting true predictor π_θ by Algorithm 1 is greater than $1-\tau$, i.e., $Pr(\pi_\theta \in \hat{\pi}_\theta) > 1-\tau$; while the probability of outputting a non-predictor t by Algorithm 1 is smaller than τ, i.e., $Pr(t \in \hat{\pi}_\theta) < \tau$ for all non-predictor t.

Proof. According to the law of large numbers (Hacking (1983)), for any positive ε_1,

$$\lim_{|\ell|\to\infty} Pr(|f_\ell(\pi_\theta,\theta) - \mu| > \varepsilon_1) = 0,$$

i.e., for any error bound τ_1, if we choose $\nu \leq \mu - \varepsilon_1$, then there exists n_1, such that for $|\ell| > n_1$, $Pr(f_\ell(\pi_\theta,\theta) > \nu) \geq Pr(f_\ell(\pi_\theta,\theta) > \mu - \varepsilon_1) \geq Pr(|f_\ell(\pi_\theta,\theta) - \mu| \leq \varepsilon_1) > 1 - \tau_1$. Then with probability larger than $1-\tau_1$, Algorithm 1 will output π_θ as a predictor. On the other hand, for a non-predictor t, the probability of t occurring prior to θ is $\mu' < \mu$ (otherwise t itself will be deemed as a predictor according to Definition 1). Then similarly, according to the law of large numbers, for any positive ε_2,

$$\lim_{|\ell|\to\infty} Pr(|f_\ell(t,\theta) - \mu'| > \varepsilon_2) = 0,$$

i.e., for any error bound τ_2, if we choose $\nu > \mu' + \varepsilon_2$, then there exists n_2, such that for $|\ell| > n_2$, $Pr(f_\ell(t,\theta) > \nu) < Pr(f_\ell(t,\theta) > \mu' + \varepsilon_2) \leq Pr(|f_\ell(t,\theta) - \mu'| > \varepsilon_2) < \tau_2$. Then with probability smaller than τ_2, Algorithm 1 will output t as a predictor. Therefore letting $\mu' < \nu < \mu$ and $n := \max(n_1, n_2)$ we have that, with probability larger than $1-\tau$ where $\tau := \min(\tau_1,\tau_2)$, Algorithm 1 outputs a correct predictor, whereas with probability smaller than τ, it outputs a non-predictor, as desired. ■

5. PATTERN MINING IN NOISE-FREE SETTING

In previous section, we proposed a general algorithm to recursively estimate the set of maximal predictors for each critical event $\theta \in \Sigma_\theta$. Note in the noise-free case, a predictor *always* precedes a critical event, and so the estimator (2) reduces to that of determining the set of maximal common subsequences of $s_1^\theta, s_2^\theta, \ldots, s_{n_\theta}^\theta$. Finding the longest common subsequence of two given sequences has been studied in the context of bit-string operations in Hunt and Szymanski (1977); Hirschberg (1975); Allison and Dix (1986). For example, Hirschberg (1975) gave quadratic complexity algorithm for finding the maximal common subsequence of two given sequences. Allison and Dix (1986) further reduced the complexity by considering the capability of a computation unit. Next in this section, we first reproduce the algorithm from Allison and Dix (1986) and then adopt it for *recursive* computation of the estimator (2) for its online use.

For each $\sigma \in \Sigma - \Sigma_\theta$ and $s \in (\Sigma - \Sigma_\theta)^*$, s_σ-trace is a binary trace of length $|s|$ where "1" indicates that the corresponding element of s^{-1} is σ. Given s and $t = \sigma_1\sigma_2\ldots\sigma_{|t|} \in (\Sigma - \Sigma_\theta)^*$, construct a $|t| \times |s|$ matrix $M_{t,s}$, such that each row is a binary vector of length $|s|$ and the ith row is given by

$$\text{row}_i = r \wedge ((r - (\text{row}_{i-1} \Leftarrow 1)) \neq r), \text{ where}$$
$$r = \text{row}_{i-1} \vee s_{\sigma_i}\text{-trace},$$

where $\neq$ denotes the non-equivalence operation and $\text{row}_{i-1} \Leftarrow 1$ is the logical left-shift of row_{i-1} with the left-most bit discarded and the right-most bit supplemented by a “1”. The $M_{t,s}$ matrix is then used to construct another matrix $\widehat{M}_{t,s}$ with size $|t| \times |s|$ and ijth element given by

$$\widehat{M}_{t,s}(i,j) := \sum_{k=1,\ldots,j} M_{t,s}(i,k).$$

Given two event sequences s and t and their $\widehat{M}_{t,s}$ matrix, the set of common subsequences of s and t, which we denote below as $C_{s,t}$, can be found by the algorithms proposed in Hunt and Szymanski (1977); Allison and Dix (1986), and which is used in Algorithm 2.

Example 2. (Reproduced from Allison and Dix (1986)). For $\Sigma - \Sigma_\theta = \{A, C, G, T\}$ and

$$s^{-1} = GTCTTACATCCGTTCG,$$

the four s_σ-traces of s are given as:

$$\begin{array}{rcccccccccccccccc}
s^{-1}: & G & T & C & T & T & A & C & A & T & C & C & G & T & T & C & G \\
s_A\text{-trace}: & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
s_C\text{-trace}: & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
s_G\text{-trace}: & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
s_T\text{-trace}: & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0
\end{array}$$

Let $t = TAGCTTAAGATCTTGT$, then the $\widehat{M}_{t,s}$ matrix is given in Fig. 3. ■

10	10	9	9	8	7	7	7	7	6	6	5	4	3	2	1	T
10	9	9	9	8	8	7	7	7	6	6	5	4	3	2	1	G
9	9	9	9	8	7	7	7	7	6	6	5	4	3	2	1	T
9	9	8	8	8	7	7	7	7	6	6	5	4	3	2	1	T
8	8	8	7	7	7	7	6	6	6	6	5	4	3	2	1	C
7	7	7	7	7	6	6	6	6	5	5	5	4	3	2	1	T
7	6	6	6	6	6	6	6	5	5	5	5	4	3	2	1	A
7	6	6	6	6	6	5	5	5	5	5	5	4	3	2	1	G
6	6	6	6	6	6	5	5	4	4	4	4	4	3	2	1	A
5	5	5	5	5	5	5	5	4	4	4	4	4	3	2	1	A
5	5	5	5	4	4	4	4	4	4	4	4	4	3	2	1	T
4	4	4	4	4	4	4	4	4	3	3	3	3	3	2	1	T
3	3	3	3	3	3	3	3	3	3	3	2	2	2	2	1	C
3	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1	G
2	2	2	2	2	2	2	2	1	1	1	1	1	1	0	0	A
1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	T
G	T	C	T	T	A	C	A	T	C	C	G	T	T	C	G	

Fig. 3. $\widehat{M}_{t,s}$ Matrix for Example 2.

Next we adopt the above algorithm for recursive computation of the estimator (2). The algorithm maintains a set of maximal predictors C_θ for each critical event θ, and which it updates only when a new critical event arrives, by comparing the existing maximals in C_θ against the trace executed following the last critical event. Then we propose the following as an estimator for the set of maximal predictors for $\theta \in \Sigma_\theta$:

$$\hat{\pi}_\theta := \{s \in C_\theta \mid \nexists t \in C_\theta, t \neq s : s \ll t\}. \quad (3)$$

Algorithm 2. Assume without of loss generality that the very first event in ℓ is a non-critical event.

```
1 Initialization:
    • $\forall \theta \in \Sigma_\theta$: $s_\theta := \epsilon$, $n_\theta := 0$;
2 Upon arrival of new $\sigma \in \Sigma - \Sigma_\theta$, do:
    • $\forall \theta \in \Sigma_\theta$: $s_\theta := s_\theta \sigma$.
3 Upon arrival of new $\theta \in \Sigma_\theta$, do:
    • update $C_\theta$ according to
      $C_\theta := \begin{cases} \{s_\theta\} & \text{if } n_\theta = 0 \\ \Big\{ t \in \bigcup_{s \in C_\theta} C_{s,s_\theta} \mid \nexists t' \in \bigcup_{s \in C_\theta} C_{s,s_\theta}, t' \neq t : t \ll t' \Big\} & \text{otherwise} \end{cases}$
    • $s_\theta := \epsilon, n_\theta := n_\theta + 1$.
```

As with Algorithm 1, the above algorithm takes as input the event set Σ, the critical event set $\Sigma_\theta \subseteq \Sigma$ and an event log ℓ, and outputs an estimator $\hat{\pi}_\theta$ for the set of maximal predictors for each $\theta \in \Sigma_\theta$.

Remark 2. As in Algorithm 1, the overall complexity for updates when a non-critical event arrives is $O(|\ell| \times |\Sigma_\theta|)$. Now for the update step when a critical event arrives, note that for $\theta \in \Sigma_\theta$, $|C_\theta| \leq m^2$, and each trace in C_θ is upper bounded in length by m; the newly arrived sequence satisfies: $|s_\theta| \leq m$, and so the complexity of this step that finds common sequences between s_θ and each trace of C_θ is $O(m^4)$. Therefore for an event log ℓ the overall complexity of for this step is $O(n \times |\Sigma_\theta| \times m^4)$, where $n := \max_\theta n_\theta$. Since $n \leq |\ell|$, it follows that the complexity for processing an event log ℓ is $O(|\ell| \times |\Sigma_\theta| \times m^4)$, where the parameter m is upper bounded by the number of events G can execute between a pair of same critical events.

6. CONCLUSION

In this paper, we studied the problem of pattern mining for predicting critical events from the observed log of ordered event data. We considered the general framework in which a predictor can be of any length, and the observations can contain interleaving from non-predictor and other predictor event sequences. This generalizes the work of Wang and Johnson (2007) who assumed that predictor-patterns must be subtraces (executed contiguously) as opposed to subsequences (need not be executed contiguously) of system traces. We proposed recursive algorithms for computing the set of maximal predictors for critical events for setting with or without noise. For the setting with noise, a critical event may not always be preceded by its predictor, and our algorithm outputs the set of “maximal” event sequences whose frequency of occurrence preceding to critical event is higher than a user-specified confidence level. We showed the existence of a lower bound on the length of an observed ordered event log that allows the discovery of a predictor with a desired accuracy. In the noise-free settings, a critical event can occur only after the execution of its predictor. We proposed an algorithm to recursively compute the set of maximal predictors for each critical event. Future work will focus on the performance analysis of the proposed mining algorithms, in terms of ability to discover predictor-patterns as measured by the probabilities of false-negatives/false-positives.

REFERENCES

Agrawal, R. and Srikant, R. (1995). Mining sequential patterns. In *Proc. 1995 Int. Conf Data Eng.*, 3–14.

Allison, L. and Dix, T.I. (1986). A bit-string longest-common-subsequence algorithm. *Inf. Proc. Lett.*, 23(5), 305–310.

Angluin, D. (1987). Learning regular sets from queries and counterexamples. *Inf. Comput.*, 75(2), 87–106.

Bailey, T.L. and Elkan, C. (1995). Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Machine Learning*, 21(1-2), 51–80.

Cabasino, M.P., Darondeau, P., Fanti, M.P., and Seatzu, C. (2011). Model identification and synthesis of discrete-event systems. *Contemporary Issues in Systems Science and Engineering.*

Carrasco, R.C. and Oncina, J. (1994). Learning stochastic regular grammars by means of a state merging method. *Grammatical Inference and Applications*, 862 of LNCS, 139–152.

Chen, J. and Kumar, R. (2013a). Decentralized failure diagnosis of stochastic discrete event systems. In *Proc. 9th IEEE Int. Conf. Autom. Sci. and Eng.*, 1083–1088. Madison, WI.

Chen, J. and Kumar, R. (2013b). Online failure diagnosis of stochastic discrete event systems. In *Proc. 2013 IEEE Multi-Conf. Syst. and Control*, 194–199. Hyderabad, India.

Chen, J. and Kumar, R. (2013c). Polynomial test for stochastic diagnosability of discrete event systems. *IEEE Trans. Auto. Sci. and Eng.*, 10(4), 969–979.

Chen, J. and Kumar, R. (2014a). Failure diagnosis of discrete-time stochastic systems subject to temporal logic correctness requirements. In *Proc. 2014 IEEE Int. Conf. Netw. Sensing, and Control.* Miami, FL.

Chen, J. and Kumar, R. (2014b). Failure prognosability of stochastic discrete event systems. In *Proc. 2014 Amer. Control Conf.* Portland, OR.

Cheng, H., Yan, X., and Han, J. (2005). Seqindex: Indexing sequences by sequential pattern analysis. In *Proc. 2005 SIAM Int. Conf. Data Mining.*

Chudova, D. and Smyth, P. (2002). Pattern discovery in sequences under a markov assumption. In *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery and Data Mining*, 153–162.

Cook, J.E. and Wolf, A.L. (1998). Discovering models of software processes from event-based data. *ACM Trans. Softw. Eng. and Methodology*, 7(3), 215–249.

Cybenko, G. and Crespi, V. (2011). Learning hidden markov models using nonnegative matrix factorization. *IEEE Trans. Inf. Theory*, 57(6), 3963–3970.

Genc, S. and Lafortune, S. (2009). Predictability of event occurrences in partially-observed discrete-event systems. *Automatica*, 45(2), 301–311.

Gold, E.M. (1967). Language identification in the limit. *Inf. Control*, 10(5), 447–474.

Gold, E.M. (1978). Complexity of automaton identification from given data. *Inf. Control*, 37(3), 302–320.

Hacking, I. (1983). Nineteenth century cracks in the concept of determinism. *Journal of the History of Ideas*, 44(3), 455–475.

Han, J., Dong, G., and Yin, Y. (1999). Efficient mining of partial periodic patterns in time series database. In *Prof. 1999 Int. Conf. Data Eng.*, 106–115. IEEE.

Hirschberg, D.S. (1975). A linear space algorithm for computing maximal common subsequences. *Comm. ACM*, 18(6), 341–343.

Hsu, D., Kakade, S.M., and Zhang, T. (2012). A spectral algorithm for learning hidden markov models. *J. Comput. Syst. Sci.*, 78(5), 1460–1480.

Hunt, J.W. and Szymanski, T.G. (1977). A fast algorithm for computing longest common subsequences. *Comm. ACM*, 20(5), 350–353.

Jiang, S. and Kumar, R. (2004). Failure diagnosis of discrete-event systems with linear-time temporal logic specifications. *IEEE Trans. Autom. Control*, 49(6), 934–945.

Jiang, S. and Kumar, R. (2006). Diagnosis of repeated failures for discrete event systems with linear-time temporal-logic specifications. *IEEE Trans. Auto. Sci. Eng.*, 3(1), 47–59.

Kumar, R. and Takai, S. (2010). Decentralized prognosis of failures in discrete event systems. *IEEE Trans. Autom. Control*, 55(1), 48–59.

Liu, C.C. and Narayan, P. (1994). Order estimation and sequential universal data compression of a hidden markov source by the method of mixtures. *IEEE Trans. Inf. Theory*, 40(4), 1167–1180.

Liu, J.S., Neuwald, A.F., and Lawrence, C.E. (1995). Bayesian models for multiple local sequence alignment and gibbs sampling strategies. *J. Amer. Stat. Asso.*, 90(432), 1156–1170.

Merhav, N., Gutman, M., and Ziv, J. (1989). On the estimation of the order of a markov chain and universal data compression. *IEEE Trans. Inf. Theory*, 35(5), 1014–1019.

Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., and Hsu, M.C. (2004). Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Trans. Knowl. Data Eng.*, 16(11), 1424–1440.

van der Aalst, W., van Dongen, B.F., Herbst, J., Maruster, L., Schimm, G., and Weijters, A. (2003). Workflow mining: a survey of issues and approaches. *Data & Knowl. Eng.*, 47(2), 237–267.

van der Aalst, W., Weijters, T., and Maruster, L. (2004). Workflow mining: Discovering process models from event logs. *IEEE Trans. Knowl. Data Eng.*, 16(9), 1128–1142.

Wang, X. and Johnson, T.L. (2007). Discovery of intermingled event patterns in discrete monitoring data. In *Proc. 1st IFAC Workshop Dependable Control of Discrete Systems*, 55–60. Pairs, France.