Slice-and-cluster: Summarization of discrete event signals patterns with event clustering on time segments

1st Anonymous
Anonymous

2nd Anonymous
Anonymous

***Abstract*—We present an approach for summarizing a given set of discrete valued time series. Our work is generalization of [1] which presents a formulation and an optimal strategy to summarize binary valued time series. Summarization, concisely represents a large event sequence emphasizing the fundamental characteristics of the entire data set. Having a global overview of the dataset allows the user to explore different options of where to start a more detailed analysis. Our approach yields summaries that 1) are concise and accurate, outputting a short and correct description of the input data. 2) provide a global data model, i.e., describe the global structure and how it evolved through the time dimension. 3) provide a local pattern model, describing local patterns so the user can identify similar behavior between events during a segment in time. and 4) is parameter-free, which means that apart from the initial event sequence, additional information need not be provided by the user. We use the Minimum Description Length (MDL) principle to detect regularities and compress data. We apply two dynamic programming algorithms to optimally and automatically segment the sequences in time and cluster events to discover similarities in polynomial time. Experiments on synthetic datasets effectively and efficiently summarize event sequences, finding correct cuts and clusters. Experiments on real datasets, including Ebola cases in Liberia, and the Global Landslide Catalog, demonstrate that our summarization finds useful patterns and tell stories validated with the analysis made by experts in the respective dataset domain.**

***Index Terms*—summarization, discrete event signals, minimum description length, dynamic programming, clustering**

I. Introduction

Complex dynamical systems, from computer systems, epidemics, to natural events, are only observable through the temporal events they emit. Quite often, actions and policies, requires an understanding of the targeted complex system. Usually, this includes, detecting changes in the frequency of event occurrences as a function of time and finding explanations for the changes. A sequence of these events consists of pairs (e, t), where e is the type of event and t is the occurrence time. Large amount of sequences of events log are being collected from different environments, like sensors, disaster management, epidemic control, etc. Iterating through different event pattern mining techniques to discover hidden temporal patterns from these logs can take time. Also, these techniques outputs involve a large number of recurring patterns infeasible for manual analysis. Before committing to a detailed analysis and the depuration of it's potentially massive results, is clever to have an overview of the entire data log. Event summarization is a process that summarizes the characteristics of the events within a system logs. In other words, event summarization emphasizes the fundamental characteristics of an entire data set. Our main objective is to provide an overview of multivariate event sequences in a glimpse. Because of this, our summarization will 1) be concise and accurate, outputting a short and correct description of the input data. 2) Provide a Global data model, describing the global structure and how it evolved through the time dimension. 3) Provide a Local pattern model, describing local patterns so the user can identify similar behavior between events during a segment in time. And 4)

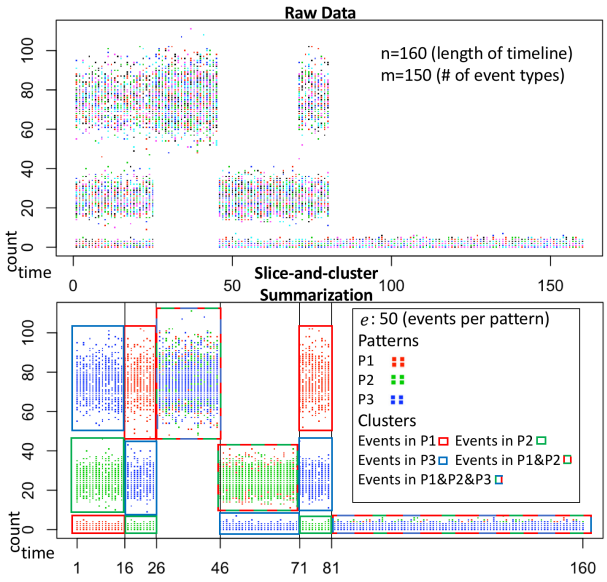

Fig. 1. The raw data in the top figure shows a timeline by day, from 1 to 160. Each dot in the raw data, represents a daily event frequency from 150 events. These events follow group patterns identified by the Slice-and-cluster summarization in the bottom figure. The output of the summarization is, first, the segmentation of the time dimension {1-15, 16-25, 26-45, 46-70, 71-80, 81-160} and second, the local clusters for each of these time segments. Each local cluster is a list of events id's that form part of the cluster and the summarization is a daily rate. Because this is synthetic data we know the list of events that conforms a pattern, and the algorithm accurately detects them. For visual aid the local clusters are represented by a color box of the pattern an event belongs to, if the events are all from different patterns but can be represented by the same daily rate the color boxes are combined.

parameter free, which means than apart from the initial event sequence, additional information should not be provided by the user. Figure 1 shows an illustrative example.

Many of the recent work in event summarization can be divided between two methodologies, frequency change [1], [2] and temporal dynamics [3]–[6]. In [1], the summarization problem is described as an optimization problem. MDL is used to find a balance between the summary's length and description's accuracy. Two dynamic programming algorithms are used to solve the problem in polynomial time. In the end it provides an overall description of the event sequences, including local similarities of the events for each segment.

In [2] the previous algorithm was extended by adding inter-segment information to describe the event generation process. Like in [1] the summarization problem is formulated as an optimization problem. It finds the best segmentation and a Hidden Markov Model (HMM) describes an event sequence with the minimum amount of information. The MDL principle is used to represent the event sequence. A HMM is used to model the state transition of a system. To solve the problem of disconnected segmentations in the summarization the HMM takes into account the correlation between the occurrence of events. A limitation of this approach is that the experiment on real data is limited, and it is not clear how high interpretability is achieved.

In [3] an event sequence is summarized using inter-arrival histograms to capture the temporal relationship among events. The temporal relationships are among same-type and different-type events, then it finds a set of disjoint histograms to summarize the input event sequence based on MDL. At the end, the resulting summary is represented as an event relationship network. It uses MDL and multi-resolution analysis for pruning the problem space. An event sequence is decomposed into many disjoint subsets and periodic patterns and correlation patterns are used to describe each subset. The inter-arrival histograms show event patterns and capture temporal dynamics of events.

These approaches [1]–[3] describe a concrete event summarization method, similar to the one we are proposing. For the frequency change approach, a segmentation model is provided. Each segment is described by a local model where the event types are grouped and clustered by frequency similarity. The temporal dynamics approach intents to reveal how the sequence changes over time. This can be done through the temporal dynamics of the segments or the temporal dynamics among the events. These approaches also deal with unique counts of events, this means that multiple events of the same type occurring at point t, are ignored as duplicates and counted as one occurrence. There is not straightforward extension for a model describing binary values (bv) event sequences to be able to use discrete values (dv) event sequences instead. Our work generalizes [1] approach for discrete-valued event sequences. This means other models like [2], [3] can use our generalization to apply their own summarization methods.

Other summarizations define their own methods for summarizing event sequences. In [4] a divide-and-conquer process is employed to extract temporal patterns based on entropy ranking and an Event Relationship Network (ERN) is constructed to provide concise representations. In [5] temporal properties are mined from logged events maintaining these invariants during summarization. It uses coarsening to explore the representations space starting with smaller representations. In [7] a sequential text clustering algorithm automatically discovers, the templates generating the messages, in system event logs. Then a second algorithm discovers patterns of messages that represent processes occurring in the system. In [6] MDL is used to identify the set of sequential patterns that summarizes the data best. It formalizes how to encode sequential data using sets of serial episodes, and uses the encoded length as a quality score. One algorithm selects a good pattern set from a large candidate set, while a second algorithm is parameter free and mines pattern sets directly from the data.

After reviewing these methods we can see how there is no problem that can be tackle by one algorithm or model. There is no optimal summarization for real world data. Also, there is no generalization for any method describing binary values event sequences to be extended to discrete values event sequences. This is why we propose an algorithm that produces an optimal high level representation of the event sequences generalizing the generative model used for bv event sequences for dv event sequences.

A. Our approach

- We present a parameter-free optimal summarization in polynomial time using MDL and Dynamic Programming.
- Our model identifies similarities between event sequences through time. Segmentation provides a high level overview describing the global structure and how it evolved through the time dimension and each of these segments will be clustered to discover local similarities between the series.
- To test the effectiveness and efficiency of our model, we design two sets of experiments on both synthetic and real datasets. By generating an artificial dataset we are capable of identifying the optimal summarization and in this way test the accuracy and limitations of our model. The summarization of a real world dataset is validated with analysis made by experts in the respective dataset domain.

B. Contributions

In summary, our contributions are:

- A novel, parameter-free, summarization-based approach to analyze event sequence data and create optimal global summaries using Minimum Description Length and Dynamic Programming.
- We generalize the generative model used for binary valued event sequences for discrete valued event sequences.
- We validated our algorithms through a synthetic dataset generator. The generated datasets have known segments and clusters. The validation step entail checking if our

proposed algorithm can pick the segments and cuts in presence of, varying magnitude, of noise
- We present summarization results on real datasets which have been analyzed and reasoned upon by experts from that domain. We show that summarization produced by our algorithm corroborates with that of domain expert's analysis.

The rest of this paper is organized as follows. Section 2 introduces the notation and defines the problem. Section 3 discusses background theory. Section 4 describes our approach for the model. Section 5 describes the optimal solution for the model and it's time complexity. Section 6 contains experiments with artificial and real datasets. Section 7 discusses related work. Section 8 presents conclusions and future work.

II. Notation and Problem Statement

A. Notation

Table I summarizes the major notations used in this and subsequent sections.

TABLE I
NOTATION

Symbol	Description
n	length of timeline
m	number of event types
$\epsilon = \{E_1, E_2, ..., E_m\}$	set ϵ of event types
S	event sequence, m x n array
$S(i,t) = x$	event E_i has x occurrences at time t
$S = \{S_1, \ldots, S_k\}$	summarized S with k segments
$M_i = \{X_{i1}, \ldots, X_{i\ell}\}$	groups of events in local model M_i
I	interval $I \subseteq [1, n]$
$S_i = S[I]$	segment defined over interval I
$n(E_j, I)$	occurrences of E_j at interval I
X_{ij}	group in M_i from segment S_i
$\lambda(X_{ij})$	average number of any event occurrence in group X_{ij}
$\lambda_d(X_{ij})$	daily average number of any event occurrence in group X_{ij}
LGM	Code length for the global model
LLM	Code length for the local model

We have a set ϵ of m event types, $\epsilon = \{E_1, E_2, ..., E_m\}$. A pair (E, t), describes that the event E occurs at time t. We have discrete timelines in the interval $[1, n]$, in which the occurrence times of events are positive integers. An event sequence can be denoted as S and records the event occurrences during the time range $[1, n]$. S can be denoted as an m x n matrix, where $S(i,t) = x$ denotes that the event E_i has a number of x occurrences at time t. Given interval $I \subseteq [1, n]$, $S[I]$ is a segment defined over the time interval I. $S_i = S[I]$ and denotes the m x $|I|$ projection of S on the interval I. The sum of the occurrences of E_j in interval I is denoted as $n(E_j, I)$.

B. Problem Statement

Given an event sequence S and records of occurrences for the events $E_i \in S$ in the timeline $[1, n]$, find integer k and a segmental grouping M of $S = \{S_i, \ldots, S_k\}$ with the best local model grouping M_i for each S_i. We want our summarization to provide a global description that 1) shows how often do the events occur. 2) Shows how the frequency of occurrences change in the course of time. 3) Find related events that might provide an explanation for the changes in frequency.

We want our model to segment the timeline and cluster event sequences based on their frequency of occurrence. Figure 2 shows the actual segmental grouping that our model finds for this event sequence. The model cuts the timeline in three segments [1,8], [9,17] and [18,30]. In the first segment, two groups consist of event sequences $\{E_1, E_2\}$ and $\{E_3\}$, respectively. In the same way the second segment groups $\{E_1, E_3, E_2\}$ and the third segment groups$\{E_1\}$ and $\{E_3, E_2\}$.

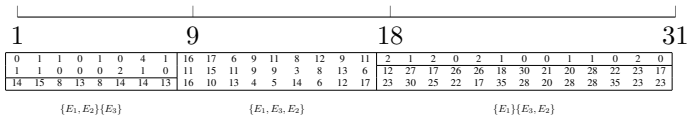

Fig. 2. Visual representation of our event sequence summarization. Here n=30 and m=3. The model cuts the timeline in [1,8] grouping $\{E_1, E_2\}\{E_3\}$. Then cuts [9,17] grouping $\{E_1, E_3, E_2\}$. Finally it cuts [18,30] grouping $\{E_1\}\{E_3, E_2\}$.

The summarization of the event sequences partition S into k segments, $S = (S_1, S_2, ..., S_k)$ where $0 < k \leq n$. After segmentations there are $k+1$ cuts, $\{c_1, c_2, ..., c_i, ..., c_{k+1}\}$ to represent the boundaries of the k segments. Here, $c_1 = 1$, $c_{k+1} = n + 1$ and the rest of the boundaries c_j take values between $2 \leq j \leq k$. A model Mi partition the rows of S_i, clustering the event types in ϵ into ℓ groups $\{X_{i1}, ..., X_{i\ell}\}$ such that $X_{ij} \subseteq \epsilon$, $X_{ij} \cap X_{ij'} = \emptyset$ for every $j \neq j'$ and $1 \leq j, j' \leq \ell$. For each group X_{ij} the parameter $\lambda(X_{ij})$ will be defined as the average number of any event part of group X_{ij} within the data segment S_i. $\lambda_{daily}(X_{ij})$ will be defined as the daily average number of any event, part of group X_{ij}, within the data segment S_i. $S(i,t) = x$ shows that event E_i has x occurrences at time t, where $x \geq 0$. In Figure 2, if E_1 represents the first row, E_2 represents the second row, and E_3 represents the third row; S(1,7)=4 and S(2,7)=1. $n(E_j, I)$ represents the number of occurrences of E_j at interval I. In Figure 2, if $I = [1, 8]$ then $n(E_1, I) = 8$ and $n(E_2, I) = 5$. Also, if $I = [1, 8]$ is represented by a local model with the following group of events, $M_1 = \{X_{11}, X_{12}\}$, with $X_{11} = \{E_1, E_2\}$ and $X_{12} = \{E_3\}$; then $\lambda_d(X_{11}) = \frac{n(E_1,I)+n(E_2,I)}{I*|X_{11}|} = \frac{13}{8*2} = 0.81$, and $\lambda_d(X_{12}) = \frac{n(E_3,I)}{I*|X_{12}|} = \frac{99}{8*1} = 12.38$.

C. Challenges

1) There is no generalization for any method describing binary values event sequences to be extended to discrete values event sequences.
2) The problem statement defines a compressing pattern problem; giving sequence S, there are multiple window conditions, and for each condition a set of candidate patterns $\mathcal{M}$ is presented. Finding an optimal model for this definition is NP-Hard [8].
3) There is no optimal summarization for real world data.

We provide solutions for these challenges in the subsequent sections, 1) Section IV-B, 2) Section V-B, 3) Section VI-B.

III. Theory

In this section we introduce the Minimum Description Length (MDL) principle.

A. MDL

The concept of MDL was defined by Rissanen [9], [10]. It has its roots in information theory and can be dated back to the concept of the Kolmogorov Complexity and the invariance theorem.

A formal language is needed to describe a data sequence. For example, a general purpose language. The Kolmogorov Complexity of a sequence, is the length of the shortest computer program that produces the initial sequence as an output. The complexity is the minimal description length. The lower the Kolmogorov complexity of a sequence the more regular it is. If there is a random sequence, the Kolmogorov complexity is close to the entropy.

Entropy: Entropy measures the uncertainty associated with a random variable. The entropy of a random variable X denoted $H(X)$ is a lower bound on the average length of the shortest description of the random variable [11]. For example the expected value of the information in the message in classical informatics is measured in bits. The Shannon entropy is an important metric in information theory. It was introduced by C. E. Shannon in [12]. Shannon entropy estimates the average minimum number of bits needed to encode a string of symbols based on an alphabet size and the frequency of the symbols. Is calculated using the following formula

$$H(X) = -\sum_{i=1}^{n} p(x_i) log_2 p(x_i).$$

The Kolmogorov complexity K is approximately equal to the Shannon entropy H if the sequence is drawn at random from a distribution that has entropy H. The concept of MDL was proposed to overcome the limitations of the Kolmogorov complexity of a dataset, uncomputability and large constants. There is no computer program, that for any dataset D, returns the shortest program that will print D. The MDL principle is an approximate estimation of the Kolmogorov complexity.

1) MDL and modeling: MDL is a method for inductive inference that provides a generic solution for the model selection problem. It is based on the observation that regularities in the data help to learn about the data. The more regularities there are in the data, the more we can compress it and the more we have learned from it. Data are viewed as codes waiting to be compressed by a model. Models are compared by their ability to compress a dataset by recognizing useful information from random noise.

Classical frequentist and Bayesian methods have the assumption that there is a true probability distribution from which the observed data was sampled, and the statistics models objective is to approximate this underlying true model. But is not possible to verify that the data is a sampled from an imagined population. MDL has a clear interpretation independent of whether or not there exists some underlying true model. For the difficult model comparison task, MDL automatically and quantitatively embodies Occam's razor. The Occam's Razor principle states that between different hypothesis the one with the fewer assumptions must be selected, in other words complex models will not be preferred over simpler models.

The idea of MDL is to represent an entire class of probability distributions as models by a single universal representative model that imitates the behavior of any model in the class [13]. MDL modeling and inference do not estimate any true data generating distribution, but it searches for good probabilities models for the data, and the best fitted model will the one that allows the shortest coding of the data.

2) Two-part code version of MDL: The simplest way to represent MDL, and how we will use it, is as a two part code that takes into account two types of cost. First, the cost of encoding the model and second, the cost of encoding the data given the model. Given a dataset D and a list of candidate models $\mathcal{M}$, it will pick the model, $M \in \mathcal{M}$ that minimizes the encoding length of the data $L(D) = L(M) + L(D|M)$.

$L(M)$ indicates the encoding length in bits of the model and $L(D|M)$ indicates the encoding length of describing the data using the model M. The equation shows that during the model selection process there is a trade-off between the complexity of the model $L(M)$ and goodness-of-fit on the data $L(D|M)$. Given a list of candidate models $\mathcal{M}$ varying in complexity. A more complex model will involve more parameters describing the model data with greater detail.

Because MDL is a generic technique, we need to define the models $\mathcal{M}$, how the models $M \in \mathcal{M}$ describe a dataset and how to obtain the code lengths in bits. This will be defined in the next section.

IV. Approach

A. The global/local method for the summarization of binary valued event sequence

The notion of a model for a given data D, broadly, has two components: a generative process G and a model m. $G(m)$ typically defines a distribution $\mathcal{D}$ over a set of data items. The cost, C, of the model is composed of the cost to represent G and m, along the with the cost of "error" in observed data D and the derived distribution $\mathcal{D}$. Now, the MDL is the minimum cost among the cost of all feasible models for D. The Kolmogorov Complexity tells the language of choice to express G and does not impact the MDL, i.e., the ratio of cost to represent a given G across any two languages is not a function of size of G, but rather bounded by some constant. This implies that the MDL under a given language is also the MDL for any other language, modulo some fixed constant. However, even when the language of G is fixed, finding MDL of any given data is intractable in general.

In [1], the authors present an approach for finding MDL where the dataset is a collection of event sequences. They propose a generative process that is a natural fit for event

sequences and a cost metric such that the model corresponding to the MDL, under the proposed generative process and cost metric, yields a intuitive summarization of the input data. Furthermore, they propose a global/local dynamic programming method that computes the optimal model, i.e., a model with MDL cost, in polynomial time. Next we describe the approach of [1] to obtain an optimal model for a given binary valued time series. We begin with a few additional notations.

a) Optimal summarization through dynamic programming: Broadly, any summarization of a given input S consists of a global model with the segments $\{S_1, \ldots, S_k\}$ and a local model M_i for each segment S_i, where $M = \{M_1, \ldots, M_k\}$. The whole model has a cost which is the value that takes to represent it, this is the code Length of the Global Model (LGM). The optimal summarization is the model with least cost for the code LGM, i.e.,

$$LGM(S) = M = arg\min_{M}\{L(M) + \sum_{i=1}^{k} L(S_i|M_i)\} \quad (1)$$

where

$$M_i = arg\min_{M_i'}\{L(M_i') + L(S_i|M_i')\}$$

and represents the code Length of the Local Model (LLM). Due to theorem 2 in [1], the above formulation can be computed using dynamic programming in polynomial time. The dynamic programming can be described through two recurrence relations – the global recurrence (LGM) and the interleaved local recurrence (LLM) equation.

$$LGM(S[1,i]) = \min_{1\le j\le i}\{LGM(S[1,j]) + LLM(S[j+1,i])\} \quad (2)$$

B. Generalization to discrete-valued event sequences

Our work generalizes the above approach for discrete-valued event sequences. This is non-trivial, primarily due to following two reasons – First, there is no straightforward extension of the generative model used for bv event sequences for dv event sequences. For bv event sequence, the generative model is a single random variable with probability equal to ρ – the mean over the number of occurrences of the events in the time segment. In other words, this is a binomial distribution with mean $t * \rho$ where t is the length of the sequence. This formulation guarantees that the value ρ is always less than 1. In addition, it can be shown that this binomial distribution happens to be the one with least cost distribution for that event-block among all the choices for binomial distribution. In contrast, for the discrete-valued model, the mean of the event sequence can be greater than 1, which makes direct application of the previous approach infeasible. Naive normalization also breaks down for similar reason.

We describe our approach for summarizing discrete-valued signals which builds upon the event summarization described in the previous section. The summarization as before will produce segments along the time domain and for each segment define a grouping over the signals such that such that signals in the same group behave "similarly" during that time segment. In order to extend the global/local dynamic programming for dv signals, the following constituents are necessary: an ordering function that given a signal-block over time period I, imposes a linear ordering of the signals in S_i. Additionally, a system of models (or distribution functions) is required for any given signal-group alongwith the metric to capture the goodness of a specific model with respect to the signal group. Finally, a mechanism to derive an optimal model among the all the possible models for the signal group is required.

We begin with our approach to define linearity over signal block $S[I]$ for any given time interval I. It can be seen as a projection from $|I|$ dimensional space to one dimensional space. The linearity function captures domain level similarity of the signals. There are several ways to define linearity over signals and over time series in general. The most commonly used similarity measures tend to capture the similarity in the trend (or movement) of the signals. However, in this work we use a different measure for similarity for dv-signals. The measure compares the sum of the values during the segment as oppose to comparing the movement over time.

Next, we define the domain of the generative models $\mathcal{M}$ for $S[I]$ alongwith with a cost fuction. The cost function captures the goodness of the model. We will then derive a mechanism to obtain an optimal cost model M from $\mathcal{M}$ for any signal-group X of $S[I]$. In our approach, any model $M(x,\lambda) \in \mathcal{M}$ is defined as an outcome of x times an event occurs in an interval with the average number of events, designated as λ.

1) Poisson model for count data: Our model focuses on the number of occurrences of an event within a fixed period. Our counts are non-negative integers. The Poisson distribution is often used for this type of data. It can provide useful insights that cannot be obtained from standard linear regression models. In the data we analyze events occur randomly and uniformly in time. These events occur with a known average. Let X be the number of events occurring in a fixed period of time I. For example, let $\sum_{E\in X_j} n(E,I)$ be the number of occurrences of the events in a cluster X_j in the interval I, then $\sum_{E\in X_j} \frac{n(E,I)}{|X_j|}$ is the average number of event occurrences. This intensity parameter λ represents the expected number of occurrences in a fixed period of time, $\lambda = E[X_j]$. We want to determine the effect of changes in event clusters on the average count.

C. Model Code Length

The local model M_i partition the rows in S_i by grouping the event types into ℓ groups $X_1, ..., X_\ell$, where $1 \le j \le \ell$. Each group X_j is described by $\lambda(X_j)$; the average number of events of any event type in X_j within the segment S_i.

To describe $\lambda_d(X_j)$ we divide $\lambda(X_j)$ by the number of days in the interval, with this the daily average number of events will be

$$\lambda_d(X_j) = \sum_{E\in X_j} \frac{n(E,I)}{|X_j|I}.$$

Because we want to model the number of events $n(E,I)$ occurring within a given time interval, for $n(E,I) = 0, 1, 2....$ we will use the Poisson distribution. Assuming independence of occurrences and event types the probability of $n(E,I)$ events occurring in X_j within the segment S_i is

$$P_\lambda(n(E,I)) = Ie^{-\lambda_d(X_j)} \frac{\lambda_d(X_j)^{\frac{n(E,I)}{I}}}{\frac{n(E,I)}{I}!}.$$

The probability of data S_i given M_i is $Pr(S_i \mid M_i)$.

$$Pr(S_i|M_i) = \prod_{j=1}^{\ell} \prod_{E \in X_j} Ie^{-\lambda_d(X_j)} \frac{\lambda_d(X_j)^{\frac{n(E,I)}{I}}}{\left(\frac{n(E,I)}{I}\right)!}$$

The number of bits to encode an event with probability p is $-log(p)$. The number of bits required to describe data S_i given model M_i is $-\log(Pr(S_i \mid M_i))$.

$$\begin{aligned} LD(S_i \mid M_i) &= -\log Pr(S_i \mid M_i) \\ &= \sum_{j=1}^{\ell} \sum_{E \in X_j} \Big[I\lambda_d(X_j) - n(E,I) \log \lambda_d(X_j) \\ &\quad + I \log \Big(\Big(\frac{n(E,I)}{I} \Big)! \Big) \Big] \end{aligned} \tag{3}$$

We want to use the optimal parameter $\lambda_d(X_j)$ that minimizes LD for the local model M_i. Each group contributes to the data cost with the following function

$$\begin{aligned} F(\lambda_d(X_j)) &= \sum_{E \in X_j} \Big(I\lambda_d(X_j) - n(E,I) \log \lambda_d(X_j) \\ &\quad + I \log \Big(\Big(\frac{n(E,I)}{I} \Big)! \Big) \Big] \end{aligned} \tag{4}$$

We can see that (4) is a function of $\lambda_d(X_j)$. To find the value of $\lambda_d(X_j)$ that minimizes $LD(S_i \mid M_i)$ we need to find the value of $\lambda_d(X_j)$ for which the first derivative of $F(\lambda_d(X_j))$ with respect to $\lambda_d(X_j)$ becomes equal to 0.

$$F'(\lambda_d(X_j)) = \sum_{E \in X_j} \Big(I - \frac{n(E,I)}{\lambda_d(X_j)} \Big)$$

The value of $\lambda_d(X_j)$ for which $F'(\lambda_d(X_j))$ is equal to 0 is

$$\lambda_d(X_j) = \sum_{E \in X_j} \frac{n(E,I)}{|X_j| I}$$

1) Bits to encode the local model: After defining how our data will be described by each model we need to define the number of bits to encode the model M_i. For every group X_j where $1 \leq j \leq \ell$ we need $m \log m$ bits to identify the ordering of the events of the proposed clusters. In the same way, we need $\ell \log m$ bits to identify the ℓ partition points on this fixed order, giving us the number of events per cluster. We also need $I * \ell \log m$ bits to describe the value of $\lambda_d(X_j)$ for each cluster. The local model cost for M_i will be

$$L(M_i) = m \log m + \ell \log m + I * \ell \log m$$

2) Bits to encode the global model: On Formula 1 we defined the code length to describe the global model as LGM. The information that we need to describe the global model M is the cuts of the segmentation that will partition S into k segments, $S = (S_1, S_2, ..., S_k)$ where $0 < k \leq n$. Because there are n possible cuts, we need $k \log n$ bits to describe the global model. We can see how MDL will try to find a balance selecting simpler models with a better fit, so $L(M) = k \log n$.

V. Detailed Optimal Solution

In this section we will define our algorithm to obtain the optimal solution, analyze the complexity of the problem and describe how we will use dynamic programming to provide polynomial time algorithms to solve the problem stated.

A. Segmentation Algorithm

The global model total description length is minimized when,

$$LGM = arg\min_{M} \{k \log n + \sum_{i=1}^{k} L(S_i|M_i)\} \tag{5}$$

where

$$M_i = arg\min_{M_i'} \{m \log m + \ell \log m + I * \ell \log m + L(S_i|M_i')\}$$

and $L(Si|M_i') = Formula(3)$ minimizing the local description length of S_i.

B. Time complexity

1) Compressing Patterns Problem: Given a sequence D, a constraint window condition, a set of candidate patterns $\mathcal{M} = \{M_1, M_2, ..., M_m\}$, find an optimal local model

$$M* = arg\min_{M} \{L(M) + L(D|M)\} \tag{6}$$

where $M* \subset \mathcal{M}$.

To solve our compressing pattern problem we need to find the optimal local model and the optimal encoding of the data D with the model M. The compressing pattern problem and the optimal local model are NP-Hard [8].

2) Optimal global model: Formula (2) is a dynamic-programming recursion that for every i where, $1 \leq i \leq n$, needs to go through every j where, $1 \leq j \leq i$, to obtain the value $LGM(S[1,j]) + LLM(S[j+1,i])$. $LGM(S[1,j])$ will be a search of an already computed value so the computation that's left to be done is $LLM(S[j+1,i])$. The time required for computing $LGM(S[1,n])$ will depend on the time to evaluate LLM on every time interval, so if this can be computed optimally, formula (2) is also optimal. The running time of formula (3) is $O(n^2 T_{LLM})$, where T_{LLM} is the running time to evaluate $LLM(S[I])$.

3) Optimal local modeling: To show how the local clustering can be solved optimally in polynomial time we will describe the second dynamic-programming algorithm. The optimal grouping assumes an ordering of the event types within a segment in decreasing order of their frequencies.

From [1], we assume that the frequencies in S are ordered so that $n(T_1, I) \geq n(T_2, I) \geq ... \geq n(T_m, I)$. Assume that the optimal local model M_i constructs ℓ groups $X_1, ..., X_\ell$. Then if $T_{j1} \in X_l$ and $T_{j2} \in X_l$ with $j_2 > j_1$, then for all $T_{j'}$'s such that $j' \in \{j_1 + 1, ..., j_2 - 1\}$ we have that $T_{j'} \in X_l$.

The local Dynamic Programming (DP) algorithm in (2) finds the optimal local model for the data segment S_i in polynomial time. The proof that one dimensional clustering can be done in polynomial time using DP is proven in [14].

VI. Experimental Evaluation

To test the effectiveness and efficiency of our model, we design two sets of experiments on both synthetic and real datasets. We evaluate our algorithm on synthetic data because there is not an optimal summarization for real world data. By generating a dataset we are capable of identifying the optimal summarization and in this way test the accuracy and constraints of our model.

For the synthetic dataset our experiments will answer the following questions: 1. Can the model effectively summarize event sequences? 2. Is the model efficiently performing the summarization? 3. Can we obtain a similar cost for the model by performing random cuts and clusters in the dataset? For the real world dataset our experiments will answer the following questions: 1. The model finds useful patterns from the real world dataset? 2. The summarization is easy to understand?

A. Synthetic Data

In order to evaluate the proposed algorithms, we shall compare its output on the synthetic data with the model used to generate the data in the first place. We know the exact cuts (and thereby the induced segments) in the timeline and the grouping of the events in each segment. With this we can define the ideal (or optimal) output for our algorithm. We will evaluate the algorithms over multiple generated datasets. Table II shows the notation used in the rest of the section to describe the experiments.

TABLE II
Experiments Notation

Symbol	Description
n	length of timeline
m	number of event types
s	number of segments
p	number of correlation patterns
d_n	degree of noise
e	number of events in one pattern

We create 6 datasets (that we call $M_1, ..., M_6$) by fixing $n = 250$, $m = 150$, $s = 10$, $p = 30$, $e = 50$. We change the noise from 0 to 0.5 incrementing by 0.1. This means that to introduce 0.1 noise to our initial dataset, each value of the set, with a probability of 10%, will be changed for a random number given by a Poisson distribution with a rate parameter of $\lambda = 5$. In each dataset 30 correlation patterns with 30 defined segments in the timeline are created with 50 events in each pattern. Each dataset is randomly ordered before executing the summarization algorithm.

Table III show the results of how many segments and clusters of events are found. For a pattern to be considered found, every event from the initial setup should be in it. Our model finds correct cuts and clusters in a dataset with 0.4% of added noise.

TABLE III
Summarization accuracy for synthetic dataset, $n = 250$, $m = 150$, $p = 20$, $c = 50$, $e = 50$

Model	d_n	No. Segments	No. Patterns
M_1	0	10/10	30/30
M_2	0.1	10/10	30/30
M_3	0.2	10/10	30/30
M_4	0.3	10/10	30/30
M_5	0.4	10/10	30/30
M_6	0.5	6/10	16/30

We will evaluate the quality of the solution produced by our algorithm by using the compression ratio, CR, defined in [1]. **Compression Ratio:** or $CR(A)$, given algorithm A is equal to the optimal total length of the data compressed with algorithm A divided by encoding the data set D as it is. In our case this means a model with n cuts in the timeline and m groups per segment. The formula is defined as,

$$CR(A) = \frac{TL(A)}{TL(D)}.$$

Figure 3 shows that for each of the models in Table III, when the percentage of noise increases the CR value increases too. This shows that our model gives a lower cost when patterns are present. Figure 5 shows this effect too. The red dot shows the optimal model value for each dataset, positioning the dot far from the best cost generated from random cuts. For the noiseless dataset the optimal code-length is far from the minimum cost model generated randomly. For the other datasets it shows that our model benefits if more patterns exists, giving a lower optimal cost for the encoded dataset. To investigate the effect of the number of patterns in a dataset, we create 7 datasets by fixing $n = 50$, $m = 150$, $s = 2$, $e = 25$. We change the percentage of patterns p from 0.1 to 0.7 incrementing by 0.1. Figure 4 shows that when the percentage of patterns increase in a dataset the compression ratio value decreases.

B. Real Data

1) Summarization of Ebola Virus Disease cases: We analyze data from an agent-based model framework developed for forecasting the 2014-2015 Ebola epidemic in Liberia [15]. We have the recorded weekly cases of Ebola infections for the 15 counties in Liberia. Figure 6 shows the cases and each county is a colored event sequence. The algorithm segments

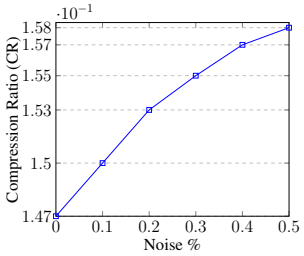

Fig. 3. When the noise increases, the compression ratio increases. Lower value of CR is preferred.

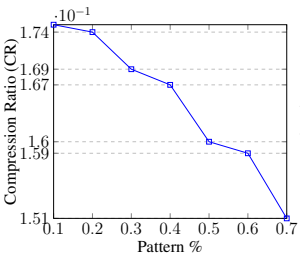

Fig. 4. When number of patterns increase, the compression ratio decreases. n=50, m=150, e=25

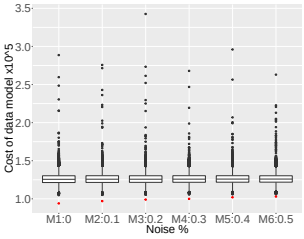

Fig. 5. For each model in Table III we generated 100,000 random models with a random assignment of cuts for segments and local clusters. Red dots represents the optimal code length for each dataset found by our optimization algorithm.

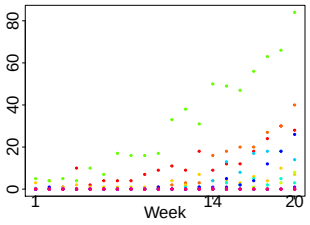

Fig. 6. Recorded weekly cases of Ebola infections for 15 counties in Liberia. Here, each county is a colored event sequence. The algorithm segments the time on week 13 and the local clusters, presented in Table IV, tell a story consistent with the analysis made by experts in the field of epidemiology.

the time on week 13 and the local clusters, presented in Table IV, tell the following story consistent with the analysis made by experts in the field of epidemiology. For the first 13 weeks Grand Cape Mount is on it's own cluster with 203 cases. The second cluster groups Bomi and Gbarpolu with the highest counts with 79 and 28. cases. The rest of counties are clustered with 9 to 0 cases for the 13 weeks. The segmentation of week 13 is important because certain practices were enforced after that week. Traditional burial practices were observed among all communities which involves washing/touching/kissing the bodies during ceremony. The practices are confirmed to contribute to the spreading of Ebola Virus Disease (EVD). No effective safe burial protocol has been enforced till week 13. Also, no contact tracing cases were followed till week 13 and no Ebola Treatment Unit (ETU) were opened till week 13.

From Table IV it is easy to see that cases are high through the 20 weeks in Grand Cape Mount and Bomi. But Gbarpolu, from having a high number of cases from week 1 to 13, descends to a low level cluster 4 in week 14.

2) Summarization of Global Landslides: The Global Landslide Catalog (GLC) was developed with the goal of identifying rainfall-triggered landslide events around the world. The GLC considers all types of mass movements triggered by rainfall, which have been reported in the media, disaster databases, scientific reports, or other sources and has been compiled since 2007 at NASA Goddard Space Flight Center

TABLE IV
SUMMARIZATION OF EBOLA VIRUS DISEASE CONFIRMED CASES IN LIBERIA, 2015.

Segment	Weeks	Clusters
1	1-13	**1:** Grand Cape Mount **2:** Gbarpolu, Bomi **3:** Bong, Montserrado, Lofa, Margibi, Grand Bassa, Grand Gedeh, Grand Kru, Maryland, Nimba, River Gee, Rivercess, Sinoe
2	14-20	**1:** Grand Cape Mount **2:** Bong, Bomi **3:** Margibi, Montserrado **4:** Gbarpolu, Lofa, Grand Bassa **5:** Nimba,Grand Gedeh, Grand Kru, Maryland, River Gee, Rivercess, Sinoe

[16]. From the GLC, we analyze almost ten years (from 01/01/2007 to 10/25/2016) of reports of fatalities by landslides. For this dataset the summarization outputs 9 segments and the corresponding local clusters are detailed in Table V.

TABLE V
SUMMARIZATION OF FATALITIES BY LANDSLIDES FROM THE GLOBAL LANDSLIDE CATALOG, FROM 01/01/2007 TO 10/25/2016.

Segment	Begin Day End Day	Begin Date End Date	Clusters
1	1 1237	01/01/2007 05/21/2010	**1:** All countries
2	1238 1313	05/22/2010 08/05/2010	**1:** China **2:** All other countries
3	1314 1315	08/06/2010 08/07/2010	**1:** China **2:** India **3:** Pakistan **4:** All other countries
4	1316 2800	08/08/2010 08/31/2014	**1:** All countries
5	2801 2823	09/01/2014 09/23/2014	**1:** India **2:** China, Pakistan **3:** All other countries
6	2824 2972	09/24/2014 02/19/2015	**1:** All countries
7	2973 2973	02/20/2015 02/20/2015	**1:** All countries
8	2974 3502	02/21/2015 08/02/2016	**1:** All countries
9	3503 3586	08/03/2016 10/25/2016	**1:** China **2:** All other countries

Segment 1 from 01/01/2007 to 05/21/2010 detects similar patterns for all countries. Segment 2 from 05/22/210 to 08/05/2010 detects, a high number of fatalities for China (2121), compared to other countries. It is interesting how segment 3, selects two days suggesting peaks in events and fatalities. During these 2 days, fatalities were only reported by China (1765), India (427) and Pakistan (58); there was one fatality reported from Trinidad and Tobago but this was clustered with the other countries with 0 reports. On 08/07/2010 in China, 1765 fatalities were reported by the cause of a landslide caused by heavy rainfall. On 08/06/2010 in India 427 fatalities

were reported by the cause of a landslide caused by flash floods due to cloud burst in Leh in Ladakh region of North India. On 08/07/2010 in Pakistan, 58 fatalities were reported by the cause of a landslide caused by heavy rainfall that raised water levels in rivers. Segment 4, detects consistently high occurrences of landslides in all countries with the peak of landslide fatalities in 2013 to the event in Kedarnath, India which killed 5669 people. Segment 5 detects a similar pattern from segment 3 where India, China and Pakistan have high occurrences of landslides from 09/01/2014 to 09/23/2014. Segment 9 detects 50 fatalities in China from 08/03/2016 to 10/25/2016. This is an interesting global pattern corroborated by previous studies [16], stating that landslide reports and landslides with fatalities peak in July through September, during the Northern Hemisphere summer, corresponding to the Southwest, South, and East Asian monsoon seasons and the Northern Hemisphere tropical cyclone peaks. Segments 6 through 8 don't show high occurrences of landslides but Segment 7 selects one day in the dataset where there are no reports from any country in the world, 02/20/2015.

C. Results and Discussion

For the synthetic dataset our model 1) effectively summarizes event sequences finding correct cuts and clusters in a dataset with 0.4% of added noise. 2) The model efficiently performs summarization decreasing the compression ratio when the number of patterns increases. 3) By performing random cuts and clusters in a dataset the resulting global model cost is higher than our optimal summarization cost. For the real world dataset our model 1) find useful patterns, for example from the GLC dataset, the global pattern corroborated that landslide reports and landslides with fatalities peak in July through September corresponding to the Southwest, South, and East Asian monsoon seasons [16]. 2) The summarization is easy to understand as we can see from Table IV and Table V.

VII. Related Work

Event summarization is closely related to sequential pattern and frequent episode mining.

A. Sequential pattern mining

The sequential pattern mining problem was first introduced by Agrawal et al [17], where Apriori based methods are used to mine frequent subsequences patterns from a sequence database. The main focus is on the patterns present on different transactions considering their sequential order. The output is frequent event subsequences with occurrence frequency bounded by a given threshold. Alternatives to the method in [17] have been proposed and can be classified in Apriori-based [18]–[20] and pattern-growth based [21], [22] algorithms. A limitation of these previous algorithms is that they don't provide a global description or summarize redundant patterns.

B. Frequent episode mining

Frequent episode mining finds temporal patterns in event sequences. An episode is defined as a collection of events that occur relatively close to each other with respect to a timeline. In [23], [24] given a window size w an episode containing an event pattern is frequent if it's frequency is bounded by a given threshold. As with the sequential pattern mining, frequent episode mining does not provide a global view of the event sequence. Also, parameters like the window size w and thresholds have to be defined.

C. Frequent item set summarization

To tackle the limitations of frequent item set mining, like redundancy and interpretability, frequent item set summarization has been proposed. The pattern profile method [25] aims to summarize the frequent patterns. K clusters contain the set of frequent patterns. Each cluster is described by a pattern profile. Based on the restoration error, a quality measure function determines the optimal value of parameter K. Also, the Markov Random Field method (MRF) [26], for frequent item set summarization, models the items in the dataset as random variables. A MRF on these variables is constructed based on frequent itemsets and their occurrence statistics. We can see how these summarization methods focus on transaction type data and usually the time information is not considered.

D. Event summarization

Even summarization methods present summaries that consider the temporal dynamics of the events. Naturally temporal, events are generally stored as logs and overlooking the time dimension has severe implications in evaluating design decisions. Different methods of event summarization can be classified between two categories frequency change [1], [2] and temporal dynamics [3]–[6]. Our model uses a frequency change approach providing a segmentation model. Each segment is described by a local model where the event types are grouped and clustered by frequency similarity. The temporal dynamics approach intents to reveal how the sequence changes over time. Our model does this through the temporal dynamics among the events in a segment and by selecting relevant segments with the minimum cost for description.

E. Minimum Description Length

The MDL approach is a generic technique that has been used in sequential pattern mining for problems like clustering [27], missing value estimation [28] and anomaly detection [29], [30]. MDL has also been used for event summarization [1]–[3], [6] but our work is the first in using MDL for generalization for any event summarization method describing bv event sequences to be extended to dv event sequences.

F. Time series segmentation

At a high level our model relates to the problem of time series segmentation [31]–[33]. The segmentation problem can be defined as the best representation given a time series T using the parameter k segments. Also, it can be defined as the best representation given a time series T such that the combined error for all the segment does not exceed some specified threshold. This work is not parameter free, for

example usually a threshold and the size of the time window needs to be defined. While some work uses MDL to segment time series by providing a parameter-free approach [34], [35] we are focusing in event sequences and not time series.

VIII. Conclusion and Future Work

This paper presents a parameter-free global summarization that creates optimal global summaries of discrete event signals patterns. We use the MDL principle to detect regularities and compress data. We apply two dynamic programming algorithms to optimally and automatically segment the sequences in time and cluster events to discover similarities in polynomial time. We generalize the generative model used for binary values event sequences for discrete values event sequences. Experiments on synthetic and real datasets demonstrate that our summarization produces high-quality results with an automated global summary revealing regularities in the data. Future work includes temporal dynamic analysis for the model output; and to include spatio-temporal characteristics in the model.

References

[1] J. Kiernan and E. Terzi, "Constructing comprehensive summaries of large event sequences," *ACM Trans. Knowl. Discov. Data*, vol. 3, no. 4, pp. 1–31, Dec. 2009.

[2] P. Wang, H. Wang, M. Liu, and W. Wang, "An algorithmic approach to event summarization," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '10. ACM, 2010, pp. 183–194.

[3] Y. Jiang, C.-S. Perng, and T. Li, "Natural event summarization," in *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, ser. CIKM '11. ACM, 2011, pp. 765–774.

[4] W. Peng, C. Perng, T. Li, and H. Wang, "Event summarization for system management," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '07. New York, NY, USA: ACM, 2007, pp. 1028–1032. [Online]. Available: http://doi.acm.org/10.1145/1281192.1281305

[5] S. Schneider, I. Beschastnikh, S. Chernyak, M. D. Ernst, and Y. Brun, "Synoptic: Summarizing system logs with refinement," in *Proceedings of the 2010 Workshop on Managing Systems via Log Analysis and Machine Learning Techniques*, ser. SLAML'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 2–2.

[6] N. Tatti and J. Vreeken, "The long and the short of it: Summarising event sequences with serial episodes," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '12. New York, NY, USA: ACM, 2012, pp. 462–470.

[7] M. Aharon, G. Barash, I. Cohen, and E. Mordechai, "One graph is worth a thousand logs: Uncovering hidden structures in massive system event logs," in *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part I*, ser. ECML PKDD '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 227–243.

[8] H. T. Lam, F. Mörchen, D. Fradkin, and T. Calders, "Mining compressing sequential patterns," *Stat. Anal. Data Min.*, vol. 7, no. 1, pp. 34–52, Feb. 2014.

[9] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, Sep. 1978.

[10] ——, "A universal prior for integers and estimation by minimum description length," *The Annals of Statistics*, vol. 11, no. 2, pp. 416–431, 1983.

[11] T. M. Cover and J. Thomas, *Elements of Information Theory*. Wiley, 1991.

[12] C. E. Shannon, "A Mathematical Theory of Communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[13] A. R. Barron, J. Rissanen, and B. Yu, "The Minimum Description Length Principle in Coding and Modeling," *IEEE Trans. Inf. Theory*, vol. 44, no. 6, pp. 2743–2760, 1998.

[14] R. Bellman, "On the approximation of curves by line segments using dynamic programming," *Commun. ACM*, vol. 4, no. 6, p. 284, June 1961.

[15] S. Venkatramanan, B. Lewis, J. Chen, D. Higdon, A. Vullikanti, and M. Marathe, "Using data-driven agent-based models for forecasting emerging infectious diseases," *Epidemics*, vol. 22, pp. 43 – 49, 2018, the RAPIDD Ebola Forecasting Challenge.

[16] D. B. Kirschbaum, R. Adler, Y. Hong, S. Hill, and A. Lerner-Lam, "A global landslide catalog for hazard applications: method, results, and limitations," *Natural Hazards*, vol. 52, no. 3, pp. 561–575, Mar 2010.

[17] R. Agrawal and R. Srikant, "Mining sequential patterns," in *Proceedings of the Eleventh International Conference on Data Engineering*, ser. ICDE '95. Washington, DC, USA: IEEE Computer Society, 1995, pp. 3–14.

[18] M. N. Garofalakis, R. Rastogi, and K. Shim, "Spirit: Sequential pattern mining with regular expression constraints," in *Proceedings of the 25th International Conference on VLDB*, ser. VLDB '99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 223–234.

[19] M. J. Zaki, "Spade: An efficient algorithm for mining frequent sequences," *Mach. Learn.*, vol. 42, no. 1-2, pp. 31–60, Jan. 2001.

[20] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu, "Sequential pattern mining using a bitmap representation," in *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '02. New York, NY, USA: ACM, 2002, pp. 429–435.

[21] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu, "Mining sequential patterns by pattern-growth: the prefixspan approach," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 11, pp. 1424–1440, Nov 2004.

[22] J. Pei, J. Han, and W. Wang, "Constraint-based sequential pattern mining: The pattern-growth methods," *J. Intell. Inf. Syst.*, vol. 28, no. 2, pp. 133–160, Apr. 2007.

[23] H. Mannila, H. Toivonen, and A. Inkeri Verkamo, "Discovery of frequent episodes in event sequences," *Data Mining and Knowledge Discovery*, vol. 1, no. 3, pp. 259–289, Sep 1997.

[24] D. Patnaik, S. Laxman, B. Chandramouli, and N. Ramakrishnan, "Efficient episode mining of dynamic event streams," in *Proceedings of the 2012 IEEE 12th International Conf. on Data Mining*, ser. ICDM '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 605–614.

[25] X. Yan, H. Cheng, J. Han, and D. Xin, "Summarizing itemset patterns: A profile-based approach," in *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, ser. KDD '05. New York, NY, USA: ACM, 2005, pp. 314–323.

[26] C. Wang and S. Parthasarathy, "Summarizing itemset patterns using probabilistic models," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '06. New York, NY, USA: ACM, 2006, pp. 730–735.

[27] K. Smets and J. Vreeken, "Slim : Directly mining descriptive patterns," pp. 236–247, 04 2012.

[28] J. Vreeken and A. Siebes, "Filling in the blanks - krimp minimisation for missing data," in *2008 Eighth IEEE International Conference on Data Mining*, Dec 2008, pp. 1067–1072.

[29] J. Vreeken, M. van Leeuwen, and A. Siebes, "Krimp: mining itemsets that compress," *Data Mining and Knowledge Discovery*, vol. 23, no. 1, pp. 169–214, Jul 2011.

[30] S. Chakrabarti, S. Sarawagi, and B. Dom, "Mining surprising patterns using temporal description length," in *Proceedings of the 24rd International Conference on VLDB*, ser. VLDB '98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 606–617.

[31] E. J. Keogh, S. Chu, D. Hart, and M. J. Pazzani, "An online algorithm for segmenting time series," in *Proceedings of the 2001 IEEE International Conference on Data Mining*, ser. ICDM '01. Washington, DC, USA: IEEE Computer Society, 2001, pp. 289–296.

[32] P. Karras, D. Sacharidis, and N. Mamoulis, "Exploiting duality in summarization with deterministic guarantees," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '07. New York, NY, USA: ACM, 2007, pp. 380–389.

[33] H. Guo, X. Liu, and L. Song, "Dynamic programming approach for segmentation of multivariate time series," *Stochastic Environmental Research and Risk Assessment*, vol. 29, no. 1, pp. 265–273, 2015.

[34] K. Xu, Y. Jiang, M. Tang, C. Yuan, and C. Tang, "Presee: an mdl/mml algorithm to time-series stream segmenting," vol. 2013, p. 386180, 06 2013.

[35] B. Hu, T. Rakthanmanon, Y. Hao, S. Evans, S. Lonardi, and E. Keogh, "Using the minimum description length to discover the intrinsic cardinality and dimensionality of time series," *Data Mining and Knowledge Discovery*, vol. 29, no. 2, pp. 358–399, Mar 2015.