# Fast and Multi-aspect Mining of Complex Time-stamped Event Streams

Kota Nakamura
SANKEN, Osaka University, Japan
kota88@sanken.osaka-u.ac.jp

Yasuko Matsubara
SANKEN, Osaka University, Japan
yasuko@sanken.osaka-u.ac.jp

Koki Kawabata
SANKEN, Osaka University, Japan
koki@sanken.osaka-u.ac.jp

Yuhei Umeda
AI Lab., Fujitsu, Japan
umeda.yuhei@fujitsu.com

Yuichiro Wada
AI Lab., Fujitsu; AIP, RIKEN, Japan
wada.yuichiro@fujitsu.com

Yasushi Sakurai
SANKEN, Osaka University, Japan
yasushi@sanken.osaka-u.ac.jp

## ABSTRACT

Given a huge, online stream of time-evolving events with multiple attributes, such as online shopping logs: *(item, price, brand, time)*, how can we summarize large, dynamic high-order tensor streams? How can we see any hidden patterns, rules, and anomalies? Our answer is to focus on two types of patterns, i.e., "regimes" and "components", over high-order tensor streams, for which we present an efficient and effective method, namely CUBESCOPE. Specifically, it identifies any sudden discontinuity and recognizes distinct dynamical patterns, "regimes" (e.g., weekday/weekend/holiday patterns). In each regime, it also performs multi-way summarization for all attributes (e.g., item, price, brand, and time) and discovers hidden "components" representing latent groups (e.g., item/brand groups) and their relationship. Thanks to its concise but effective summarization, CUBESCOPE can also detect the sudden appearance of anomalies and identify the types of anomalies that occur in practice.

Our proposed method has the following properties: (a) *Effective:* it captures dynamical multi-aspect patterns, i.e., regimes and components, and statistically summarizes all the events; (b) *General:* it is practical for successful application to data compression, pattern discovery, and anomaly detection on various types of tensor streams; (c) *Scalable:* our algorithm does not depend on the length of the data stream and its dimensionality. Extensive experiments on real datasets demonstrate that CUBESCOPE finds meaningful patterns and anomalies correctly, and consistently outperforms the state-of-the-art methods as regards accuracy and execution speed.

## CCS CONCEPTS
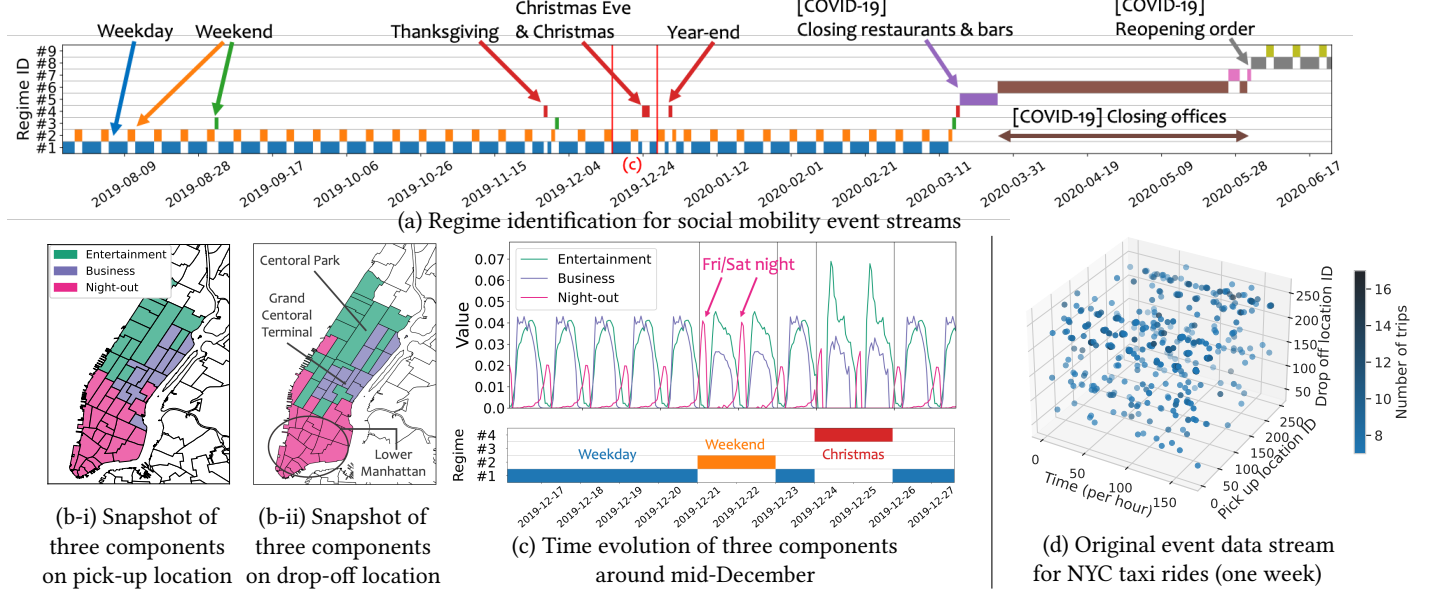
• **Information systems** → **Data mining**.

## 1 INTRODUCTION

Given a large, online stream of time-stamped events, how can we statistically summarize all the event streams and find important patterns, rules, and anomalies? Time-stamped event data are generated and collected by many real applications [10, 17, 29, 84], including online marketing analytics [52, 75], social network/location-based services [28, 71], and cybersecurity systems [19, 80], with increasingly larger sizes and faster rates of transactions. For example, an online shopping service could generate millions of logging entries every second, with rich information about items and users. The service providers would like to send targeted advertisements and detect fraudulent activities by investigating online purchasing patterns and hidden user/item relationships.

Here, let us assume that we have a large collection of event logs, consisting of multiple attributes, e.g., online shopping: *(item, price, brand, time)* and local mobility activities: *(pick-up and drop-off locations, time)*, where huge numbers of event entries arrive online at high bit rates, which we shall refer to as "*complex time-stamped event streams*". These data are represented as high-order tensor streams, e.g., a 4th-order item-price-brand-time tensor stream, unlike a previously considered multivariate time series [37], tensor [50], or stream of elements [61], specifically, as mentioned later in Section3.1, whose high-dimensional, sparse, and semi-infinite nature derails existing methods and even our interpretation of data. *So what is a good representation of complex time-stamped event streams?* This is exactly the problem we focus on in this work. We first present a compact yet powerful representation that summarizes a semi-infinite collection of tensor streams. Specifically, we aim to capture two types of patterns, i.e., "*regimes*" and "*components*".

In practice, real-life data streams contain various types of distinct temporal dynamical patterns of different durations, namely, "*regimes*", such as the weekday/weekend/holiday patterns of online shopping services or taxi rides. In each regime, a set of events, consisting of multiple attributes, has similar behavior and latent interactions. We introduce the concept of latent "*components*", which capture hidden groups in each attribute (e.g., item groups and typical pick-up locations) and their relationships.

An important application scenario, for example, in cybersecurity, multiple types of intrusions/anomalies, such as denial of service or port scanning attacks, occur suddenly and need to be detected and analyzed in real-time to minimize harm. So we would also like to answer the question: *How can we quickly detect anomalies and identify their types?* However, it is extremely challenging because signs of the anomalies appear in one or more attributes (e.g., source

Kota Nakamura, Yasuko Matsubara, Koki Kawabata, Yuhei Umeda, Yuichiro Wada, and Yasushi Sakurai



(a) Regime identification for social mobility event streams

(b-i) Snapshot of three components on pick-up location

(b-ii) Snapshot of three components on drop-off location

(c) Time evolution of three components around mid-December

(d) Original event data stream for NYC taxi rides (one week)

**Figure 1: Real-time modeling of CubeScope on New York City taxi rides: (a) It incrementally identifies distinct time-evolving patterns (i.e., regimes) and their shifting points. Specifically, Regime #1 (blue) coincides with weekdays, while Regimes #2, #3, and #4 (orange, green, and red) capture weekends and public holidays. Also, it can adaptively recognize the sudden regime transitions (Regimes #5, #6, ⋯) that reflect social conditions under the COVID-19 pandemic. It finds components that are interpretable summaries for all attributes (i.e., pick-up, drop-off, time), especially for (b) pick-up/drop-off locations and (c) time attribute. (d) The original data is a sparse and high-dimensional tensor. It exhibits no obvious components or regimes.**

IP address, packet size,. . .), and event streams evolve over time, where new types of anomalies can arise and the concept of normal behavior changes.

In this paper, we present CubeScope, an efficient and effective mining approach capable of dealing with the above questions. CubeScope monitors a high-order tensor stream and incrementally recognizes dynamical multi-aspect patterns, i.e., regimes and components, and anomalies, while updating the information for each. Intuitively, the problem we wish to solve is as follows:

InformalProblem 1. **Given** *a high-order tensor stream* $\mathcal{X}$*, which consists of events with multiple attributes and timestamps,*
- **Find** *a compact description of* $\mathcal{X}$ *that summarizes all events,*
  - *distinct dynamical patterns (i.e., regimes),*
  - *multi-aspect latent trends (i.e., components),*
- **Report** *anomalies and their types*
*incrementally and quickly, at any point in time.*

**Preview of Results.** Figure 1 (a)-(c) shows some of our discoveries on local mobility data. This dataset consists of taxi ride events *(pick-up location ID, drop-off location ID, time)* in New York City, with hourly timestamps, from Jul. 1st, 2019, to Jun. 30th, 2020. Figure 1 (d) shows the original data. The data are represented as the stream of the 3rd-order tensor, where each aspect indicates each attribute. Note that this tensor is sparse and high-dimensional, i.e., there are numerous dimensions in each aspect/attribute. It does not exhibit any obvious patterns, neither regimes nor components.

- *Regime identification:* As shown in Figure 1 (a), CubeScope incrementally discovers nine regimes (i.e., distinct time-evolving patterns). Specifically, it finds Regimes #1 (blue) and #2 (orange), corresponding to weekdays and weekends, respectively. Around the end of the year, it recognizes new regimes,

Regimes #3, #4 (green, red), which coincided with certain festive days, including Thanksgiving, Christmas, and Year-end. The new Regimes #5 (purple) and #6 (brown) indicate abrupt changes in human movement. In fact, due to the emergence of a new viral pandemic, COVID-19, the city ordered restaurants/bars to close on March 16th [11] and then offices to close on March 22nd [12]. Finally, our method generates Regimes #8 (gray) and #9 (dark yellow) for the new weekday and weekend human mobility patterns, respectively, after the reopening order on June 8th [13], which allowed office-based workers and in-store retail shopping to resume.

- *Multi-aspect component analysis:* CubeScope provides components that are interpretable summaries for each attribute. Figure 1 (b) shows the three major components for pick-up/drop-off locations in Regime #1, where we manually named them "Entertainment", "Business", and "Night-out". These areas agree with our intuition: the Entertainment component is allocated around Central Park and nearby museums, the Business component is concentrated on major railway stations such as Grand Central Terminal, and the Night-out component corresponds to the area around Lower Manhattan, which has a large number of restaurants and bars. Figure 1 (c) shows three major components for time attribute around mid-December. They show the spiking of the Entertainment component during Christmas. The Business component consistently exhibits high peaks on weekdays, while it had lower value on weekends and Christmas. Lastly, the Night-out component shows midnight peaks, especially on weekends (i.e., Fri/Sat midnight).

**Contributions.** The main contributions of our paper are:

**Table 1: Capabilities of approaches.**

| | TICC/++ | CubeMarker | T-LSTM | LDA/NTM/++ | TriMine | DBSTREAM/++ | LOF/++ | MemStream | CubeScope |
|---|---|---|---|---|---|---|---|---|---|
| High-dimensional Tensor | - | some | - | - | ✓ | - | - | ✓ | ✓ |
| Sparsity | - | - | ✓ | ✓ | ✓ | - | - | ✓ | ✓ |
| Semi-infinite Data | - | - | - | - | - | ✓ | - | ✓ | ✓ |
| Segmentation | ✓ | ✓ | ✓ | - | - | - | - | - | ✓ |
| Data Compression | ✓ | ✓ | - | ✓ | ✓ | ✓ | - | - | ✓ |
| Anomaly Detection | - | - | - | - | - | - | ✓ | ✓ | ✓ |
| Dynamical Multi-aspect Patterns | - | - | - | - | - | - | - | - | ✓ |

- **Effective**: We introduce dynamical multi-aspect patterns (i.e., regimes and components), which summarize high-order tensor streams and provide interpretable representations. Also, we formulate the summarization problem for capturing these patterns in a data compression paradigm.
- **General**: To solve the summarization problem, we design CubeScope, which also performs data compression, pattern discovery, and anomaly detection. Our experimental results show its practicality on multiple domains, such as online marketing analytics and cybersecurity.
- **Scalable**: Our proposed algorithm is fast and requires constant computational time both with regard to the entire stream length and the dimensionality for each attribute.

**Reproducibility.** Our source code and datasets are available at [6].
**Outline.** The rest of this paper is organized as follows. We first introduce related studies followed by our proposed model and algorithms, experiments and conclusions.

## 2 RELATED WORK

The mining of time-stamped event data has attracted great interest in many fields [30, 42, 46, 58, 66, 67, 69, 78, 82, 86–88]. Table 1 illustrates the relative advantages of our method, and only CubeScope meets all the requirements.

**Modeling Dynamics and Segmentation.** Classical approaches such as linear dynamical systems (LDS), and hidden Markov models (HMM) are extended to capture distinct patterns of sequences as described in [40, 48, 54, 83]. TICC [37] characterizes the interdependence between multivariate observations based on a Markov random field. Such distinct time series patterns also enable us to perform anomaly detection and forecasting [26, 63–65]. Tensor-based approaches for time series segmentation have been proposed [39, 47] that incorporate latent relationships between sequences. The previous studies are designed for continuous time series and are thus incapable of modeling sparse tensors. In recent years, many deep neural network models have been proposed [53, 59]. T-LSTM [17] identifies disease progression patterns with irregular time intervals. Since these are mostly "black-box" models and incur high computation costs, they cannot address streaming data summarization. Moreover, none of the above studies focuses on large and sparse tensors with a higher order than 3.

**Summarization and Clustering.** Probabilistic generative models [18, 71], such as latent Dirichlet allocation (LDA) [22] and its variants [43, 70, 89], are broadly applied to analyze large collections of categorical data. More recently, topic models have been extended to neural-based models [55, 85] by using a variational autoencoder [49]. A collection of events can be turned into a tensor [41, 44, 76]. TriMine [68] summarizes an event tensor and discovers groups of dimensions. As with [16, 51, 67, 78], the minimum description length (MDL) principle [33] is applied to summarize time series and dynamic graphs. Unlike these methods, our work focuses on tensor streams. The processing and clustering of data streams [14, 32, 36, 60], such as DBSTREAM [36], have also attracted significant interest. However, these algorithms process each data point individually and cannot capture multi-aspect features.
**Anomaly Detection.** Typical anomaly detection methods [15, 25, 31, 61, 81], such as local outlier factor (LOF) [24] and tree-based approaches [34, 56], can be used in event tensors by converting multiple attributes to numerical ones. [20, 45, 62, 79, 80] use a stream of multi-aspect records as input. MemStream [21] can learn dynamically changing trends to handle time-varying data distribution known as *concept drift* [27, 35, 57]. Although these methods have the ability to detect multiple anomalies, they cannot identify the types of anomalies or capture dynamical multi-aspect patterns.

In conclusion, none of the existing methods focus specifically on modeling of dynamical multi-aspect patterns, summarization, and anomaly detection in high-order tensor streams.

## 3 PROPOSED MODEL

In this section, we present our proposed model.

### 3.1 Design Philosophy of CubeScope

The symbols used in this paper are described in Appendix A. Here we consider our settings, namely, complex time-stamped event streams. We continuously monitor an event entry with $M$ categorical attributes and a timestamp. At the most recent time $T$, we have a collection of events with $U_1 \ldots U_M$ unique units for each attribute and $T$ timestamps.

**Definition 1 (Event tensor stream).** Let $\mathcal{X} \in \mathbb{N}^{U_1 \times \cdots \times U_M \times T}$ be an $M+1$ th-order tensor stream up to the current time point $T$. At every time point $T$ that is arrived at with a non-overlapping time interval $\tau \ll T$, we can obtain the current tensor $\mathcal{X}^C \in \mathbb{N}^{U_1 \times \cdots \times U_M \times \tau}$ as the partial tensor of $\mathcal{X}$. The element $x_{u_1 \ldots u_M, t}$ of $\mathcal{X}$ shows the total number of event entries of the $u_1$-th $\ldots u_M$-th units in each attribute at time tick $t$.

Figure 1 (d) shows the event tensor stream for NYC taxi rides, where each event is of the form *(pick-up ID, drop-off ID, time)*, $M = 2$. Here, we provide the reader with three important observations.

**Observation 1 (High Dimensional).** *This tensor has a large number of units in each attribute, e.g., $U_1 = 262$ and $U_2 = 264$ in the pick-up/drop-off location attribute.*

**Observation 2 (Sparse).** *In the figure, most of the attribute pairs have very sparse sequences, which derails all typical time series analysis tools because they look like noise (e.g., $\{0, 0, 0, 1, 0, 0, 1, 2, 0, \cdots\}$).*

**Observation 3 (Semi-infinite).** *The event tensor stream evolves over time and arrives in an unbounded stream, making it impossible to store all the historical data.*

Consequently, we aim to summarize the event tensor stream and obtain a succinct description. Specifically, we focus on the two types of patterns, (**P1**) components (i.e., latent groups and their

relationship) and (**P2**) regimes (i.e., distinct time-evolving patterns). So, what is the simplest mathematical model that can capture both (**P1**) and (**P2**)? How can we formulate the summarization problem? We provide the answers below.

## 3.2 Proposed Solution: CubeScope

We now present our model in detail. We first describe (**P1**) components in each current tensor $\mathcal{X}^C$ by introducing *multi-aspect component factorization* and then propose a *compact description* for representing (**P2**) regimes and the whole tensor stream. Finally, we formalize the problem as minimizing *encoding cost* in the data compression paradigm.

*3.2.1 Multi-aspect Component Factorization (P1).* We begin with the simplest case, where we have only a current tensor $\mathcal{X}^C$. Our first step is to describe a high-dimensional and sparse tensor $\mathcal{X}^C$ as a compact and interpretable model. We thus propose a new factorization to model the generative process of events. In our model, we assume that there are $K$ major trends/components behind the event collections. Specifically, the $k$-th component is characterized by probability distributions in terms of $M$ attributes and time, which are defined as follows:

- $\mathbf{A}_k^{(m)} \in \mathbb{R}^{U_m}$: probability distribution over $U_m$ units of the attribute $m$ for the component $k$.
- $\mathbf{B}_t \in \mathbb{R}^K$: probability distribution over $K$ components for the time $t$.

Here, we refer to $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(M)}$, and $\mathbf{B}$ as component matrices. Since we treat each attribute as categorical, the component matrices can be described by employing a Dirichlet prior [22]:

$$\mathbf{A}_k^{(m)} \sim \text{Dirichlet}(\alpha^{(m)}), \ \mathbf{B}_t \sim \text{Dirichlet}(\beta),$$

where $\alpha^{(m)}$ and $\beta$ are the hyperparameters[1].

We also incorporate temporal dependencies into this model so that each component matrix captures the context of its predecessors in the data stream. We assume that the means of the components are the same as at the previous time $T - \tau$, unless the newly arrived events $\mathcal{X}^C$ are confirmed. With this assumption, we can use the following Dirichlet priors: $\text{Dirichlet}(\alpha^{(m)}{}_1\hat{\mathbf{A}}_k^{(m)})$ and $\text{Dirichlet}(\beta_1\hat{\mathbf{B}}_t)$, where $_l\hat{\mathbf{A}}^{(m)}$ and $_l\hat{\mathbf{B}}$ are the previous component matrices at $T - l\tau$. To capture the long-term dependencies, we can extend this approach so that it can depend on past $L$ matrices.

$$\mathbf{A}_k^{(m)} \sim \text{Dirichlet}(\Sigma_{l=1}^L \alpha^{(m)}{}_l\hat{\mathbf{A}}_k^{(m)}), \ \mathbf{B}_t \sim \text{Dirichlet}(\Sigma_{l=1}^L \beta_l\hat{\mathbf{B}}_t).$$

Consequently, the generative process can be described as follows:

- For each component $k = 1, \ldots, K$:
  - For each attribute $m = 1, \ldots, M$:
    * $\mathbf{A}_k^{(m)} \sim \text{Dirichlet}(\Sigma_{l=1}^L \alpha^{(m)}{}_l\hat{\mathbf{A}}_k^{(m)})$
- For each time $t = 1, \ldots, \tau$:
  - $\mathbf{B}_t \sim \text{Dirichlet}(\Sigma_{l=1}^L \beta_l\hat{\mathbf{B}}_t)$
  - For each entry $j = 1, \ldots, N_t$:
    * $z_{t,j} \sim \text{Multinomial}(\mathbf{B}_t)$ // Draw a latent component $z_{t,j}$
    * For each attribute $m = 1, \ldots, M$:
      · $e_{t,j}^{(m)} \sim \text{Multinomial}(\mathbf{A}_{z_{t,j}}^{(m)})$, // Draw a unit in each attribute

where $N_t$ is the total number of events at time $t$, and $z_{t,j}$ is the latent component. Each event $e_{t,j}$ is sampled from the component-specific

multinomials. We note that the benefits of this model are three-fold. *First*, even though the event tensor has sparse activity, our model can discard many redundancies (e.g., noise) and summarize a set of events into $K$ components. *Second*, an event entry is generated from $M + 1$ component matrices. It thus handles arbitrary-order tensors. *Third*, to capture temporal dependencies, it employs past $L$ component matrices rather than storing tensors.

*3.2.2 Compact Description (P2).* Although the component matrices concisely describe the partial tensor $\mathcal{X}^C$, it is insufficient for the whole tensor stream $\mathcal{X}$, containing various types of distinct dynamical patterns. We thus introduce another higher-level architecture.

DEFINITION 2 (REGIME: $\theta$). *Let $\theta$ be a regime consisting of the component matrices: $\theta = \{\{\mathbf{A}^{(m)}\}_{m=1}^M, \mathbf{B}\}$ to describe a certain distinct dynamical pattern with which we can divide and summarize the entire tensor stream into segments. When there are $R$ regimes, a regime set is defined as $\Theta = \{\theta_r\}_{r=1}^R$.*

Also, when there are $G$ switching positions, the regime assignments are defined as $\mathcal{S} = \{s_g\}_{g=1}^G$, where $s_g = (t_s, r)$ is the history of each switching position $t_s$ to the $r$-th regime. Finally, we adopt all the above parts for a compact description of $\mathcal{X}$.

DEFINITION 3 (COMPACT DESCRIPTION). *Let $C = \{R, \Theta, G, \mathcal{S}\}$ be a compact representation of the whole tensor stream $\mathcal{X}$, namely,*

- *the number of regimes $R$ and the regime set, $\Theta = \{\theta_r\}_{r=1}^R$,*
- *the number of segments $G$ and the assignments, $\mathcal{S} = \{s_g\}_{g=1}^G$.*

*3.2.3 Problem Formulation.* Our final goal is to formulate the problem, where we summarize all data streams $\mathcal{X}$ into a compact representation $C$. Our objective function leverages the minimum description length (MDL) principle [33]. In short, it follows the assumption that the more we can compress the data, the more we can learn about their underlying patterns. Specifically, we evaluate the total encoding cost, which can be used to losslessly compress the original tensor stream $\mathcal{X}$. The summarization problem is written as follows:

PROBLEM 1. ***Given*** *a whole event stream $\mathcal{X}$,* ***find*** *the compact description $C$, which minimizes the total encoding cost*

$$< \mathcal{X}; C > = < C > + < \mathcal{X}|C >, \tag{1}$$

*where $< C >$ is the model coding cost of $C$, and $< \mathcal{X}|C >$ is the data coding cost given the model $C$.*

**Model Coding Cost.** The model coding cost is the number of bits needed to describe the model. In our model, the dimensionality of latent components requires $< d > = \Sigma_{m=1}^M \log^*(U_m) + \log^*(\tau) + \log^*(K)$ [2]. The number of regimes needs $< R > = \log^*(R)$. The model coding cost of each regime $\theta$ consists of the following terms,

$$< \theta > = \sum_{m=1}^M < \mathbf{A}^{(m)} > + < \mathbf{B} >, \tag{2}$$

$$< \mathbf{A}^{(m)} > = |\mathbf{A}^{(m)}| \cdot (\log(K) + \log(U_m - 1) + c^F) + \log^*(|\mathbf{A}^{(m)}|), \tag{3}$$

$$< \mathbf{B} > = |\mathbf{B}| \cdot (\log(\tau) + \log(K - 1) + c^F) + \log^*(|\mathbf{B}|), \tag{4}$$

where $|\cdot|$ describes the number of non-zero elements in each of the matrices, and $c^F$ is the floating point cost [3]. The number

---

[1]We set $\alpha^{(m)} = \beta = 1/K$ as default.

[2]Here, $\log^*$ is the universal code length for integers.

[3]We set 8 bits as the default by following [66, 82].

of segments needs $< G >= \log^*(G)$. Each shifting point needs $< s_g >= \log^*(t_s) + \log(R)$.

**Data Coding Cost.** Given a full regime set $\Theta$, we can encode the data $\mathcal{X}$ based on Huffman coding [23], i.e, a number of bits are assigned to each value in $\mathcal{X}$. The data coding cost of $\mathcal{X}$ given $\theta$ is computed by: $< \mathcal{X}|\theta >= -\log P(\mathcal{X}|\{\mathbf{A}^{(m)}\}_{m=1}^M, \mathbf{B})$. Thus, the data coding cost of $\mathcal{X}$ given $C$ is computed by:

$$< \mathcal{X}|C > = \sum_{r=1}^R -\log P(\mathcal{X}[r]|\theta_r), \qquad (5)$$

where $\mathcal{X}[r]$ is a set of partial tensors assigned by the $r$-th regime. Finally, the total encoding cost $< \mathcal{X}; C >$ is written as follows:

$$
\begin{aligned}
< \mathcal{X}; C > =\; & < C > + < \mathcal{X}|C > \\
=\; & < d > + < R > + < G > \\
& + \sum_{r=1}^R < \theta > + \sum_{g=1}^G < s_g > + < \mathcal{X}|C > .
\end{aligned} \qquad (6)
$$

## 4 STREAMING ALGORITHM

Thus far, we have described how we represent the two concepts, (i.e., components and regimes), and formulate the summarization problem (i.e., Problem 1) in a lossless compression context. Our next goal is to solve the problem in a streaming setting. In this section, we aim to figure out *how to incrementally summarize* entire event streams into a compact description $C$ and also *how to exploit* the compact description for streaming anomaly detection.

To tackle these problems, we now present a streaming algorithm CubeScope, consisting of two sub-algorithms, C-Decomposer and C-Compressor. Algorithm 1 (see Appendix B) shows the overall procedure, and Figure 2 illustrates how the proposed algorithm works. Intuitively, our algorithm continuously generates a regime $\theta_c$ from the non-overlapping arrival tensor $\mathcal{X}^C$. It then updates the compact description $C$ with $\theta_c$ and measures the anomalousness of $\mathcal{X}^C$. Next, we describe each sub-algorithm in detail.
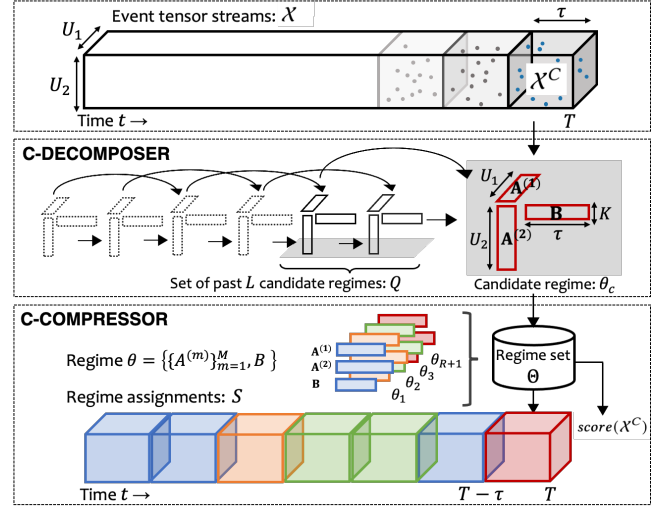
### 4.1 C-Decomposer

We first aim to incrementally monitor $\mathcal{X}^C$ and estimate a candidate regime $\theta_c$ (i.e., $\{\mathbf{A}^{(m)}\}_{m=1}^M, \mathbf{B}$), which best describes $\mathcal{X}^C$. According to the generative process in section 3.2.1, we propose an efficient estimation with collapsed Gibbs sampling [73]. Specifically, for each non-zero entry $x_{u_1,...,u_M,t}$ in $\mathcal{X}^C$, we draw latent components $z_{u_1,...,u_M,t}$ with the probability $p$:

$$p(z_{u_1,...,u_M,t} = k \mid \mathcal{X}^C, \mathbf{B}', \hat{\mathbf{B}}, \beta, \{\mathbf{A}^{(m)'}, \hat{\mathbf{A}}^{(m)}, \alpha^{(m)}\}_{m=1}^M)$$

$$\propto \frac{b'_{t,k} + \sum_{l=1}^L \beta_l \hat{b}_{t,k}}{\sum_{k=1}^K b'_{t,k} + L\beta} \cdot \prod_{m=1}^M \frac{a^{(m)'}_{k,u_m} + \sum_{l=1}^L \alpha^{(m)}{}_l \hat{a}^{(m)}_{k,u_m}}{\sum_{u=1}^{U_m} a^{(m)'}_{k,u_m} + L\alpha^{(m)}}, \qquad (7)$$

where $a^{(m)}_{k,u_m}$ and $b_{t,k}$ are the total counts of that component $k$ is assigned to the $u_m$-th unit and time-tick $t$, respectively. Note that the prime (e.g., $b'_{t,k}$) indicates the count yielded by excluding the entry $x_{u_1,...,u_M,t}$. After the sampler has burned-in, we can obtain the component matrices for $\mathcal{X}^C$, i.e., $\{\tilde{\mathbf{A}}^{(m)}\}_{m=1}^M$ and $\tilde{\mathbf{B}}$, as follows:

$$\tilde{\mathbf{A}}^{(m)}_{k,u_m} \propto \frac{a^{(m)}_{k,u_m} + \sum_{l=1}^L \alpha^{(m)}{}_l \hat{a}^{(m)}_{k,u_m}}{\sum_{u=1}^{U_m} a^{(m)}_{k,u_m} + L\alpha^{(m)}}, \quad \tilde{\mathbf{B}}_{t,k} \propto \frac{b_{t,k} + \sum_{l=1}^L \beta_l \hat{b}_{t,k}}{\sum_{k=1}^K b_{t,k} + L\beta}. \qquad (8)$$



Figure 2: Illustration of CubeScope algorithm. C-Decomposer: Given a current tensor $\mathcal{X}^C \in \mathbb{N}^{U_1 \times U_2 \times \tau}$, it first decomposes $\mathcal{X}^C$ into a candidate regime $\theta_c$ depending on past $L$ regimes ($L = 2$ in this figure). C-Compressor: it assigns the optimal regime for $\mathcal{X}^C$ among the candidate regime $\theta_c$ and the regimes $\{\theta_r\}_{r=1}^R (= \Theta)$. If the candidate regime is assigned, CubeScope inserts it into the regime set $\Theta$ as $\theta_{R+1}$. It also computes the anomalousness score for $\mathcal{X}^C$.

Algorithm 2 (see Appendix B) shows C-Decomposer in detail. It first assigns the latent component $z_{u_1,...,u_M,t}$ for each entry $x_{u_1,...,u_M,t}$ in $\mathcal{X}^C$ with Equation (7). Once the latent components are determined, we can compute the objective matrices simply with Equation (8). Here, we keep past $L$ component matrices in the past parameter set, i.e., a FIFO queue $Q$ with size $L$. After computing the current component matrices, the oldest matrices are removed, and the estimated matrices are inserted into $Q$.

Even though our tensor data is high-dimensional (Observation 1), C-Decomposer does not depend on dimensionality, i.e., it takes $O(N)$ time, where $N$ is the number of events (see Lemma 4.1 for details). In contrast, conventional tensor algorithms such as alternating least squares (ALS) [50] scale with respect to all the attributes, i.e., take $O(\prod_{m=1}^M U_m)$ time, which may become very computationally demanding for high-order tensors ($M \geq 4$).

### 4.2 C-Compressor

We next describe C-Compressor in steps. Algorithm 3 (see Appendix B) shows the overall procedure. After obtaining the candidate regime $\theta_c$, our next goal is to find compact description $C$ for the whole tensor stream $\mathcal{X}$. However, $\mathcal{X}$ is semi-infinite (Observation 3), and we thus cannot process all historical data. To efficiently find the compact description $C$, we adopt an insertion-based algorithm, where it maintains a reasonable description for $\mathcal{X}$ and generates a regime if necessary. Specifically, the algorithm tracks only two regimes, the previous regime $\theta_p$ and the candidate regime $\theta_c$. Given a current tensor $\mathcal{X}^C$ with the two regimes, the algorithm compares the extra cost according to Equation (6) for each regime, and then chooses the next procedure so that the additional cost is minimized:

- If C-Compressor uses $\theta_p$, it stays in the previous regime.

- If $\theta_c$ is chosen, C-COMPRESSOR first finds a more suitable regime in $\Theta$ to avoid duplication. Then, the least expensive regime is adopted.

Here, the additional cost $< \mathcal{X}^C; \theta_* >$ is written as follows:

$$< \mathcal{X}^C; \theta_* > = \Delta < C > + < \mathcal{X}^C | \theta_* >, \quad (9)$$
$$\Delta < C > = \log^*(R+1) - \log^*(R) + < \theta_* >$$
$$+ \log^*(G+1) - \log^*(G) + < s >, \quad (10)$$

where $\theta_*$ indicates any regime. If we need to shift another existing regime to represent $\mathcal{X}^C$, then $\Delta < C > = \log^*(G+1) - \log^*(G)+ < s >$; if the description of $\mathcal{X}^C$ requires new regimes, it costs all of the terms in Equation (10); otherwise, $\Delta < C > = 0$.

**Online Regime Updates.** Whenever an existing regime is selected, each count in the existing regime is updated by adding $\theta_c$:

$$\tilde{\mathbf{A}}_{k,u_m}^{(m)} \leftarrow \frac{a_{k,u_m}^{(m)} + \sum_{l=1}^{L} \alpha^{(m)}{}_l \hat{a}_{k,u_m}^{(m)} + {}_c a_{k,u_m}^{(m)}}{\sum_{u=1}^{U_m} a_{k,u_m}^{(m)} + L\alpha^{(m)} + \sum_{u=1}^{U_m} {}_c a_{k,u_m}^{(m)}}, \quad (11)$$

where $c$ (e.g., ${}_c a_{k,u_m}^{(m)}$) is a count in the candidate regime. $\tilde{\mathbf{B}}_{t,k}$ is analogous, and omitted for brevity. The effect of the candidate regime is decayed as the existing regime is updated; in other words, each regime converges as it updates.

**Anomaly Detection.** Finally, we exploit the compact description $C$ for anomaly detection. Compression-based techniques are naturally suited for anomaly and rare instance detection. In a given compact description $C$, the high usage regime compresses the majority of all past data with a short code length. In other words, it represents the *norm* in the data stream and thus it needs long code length against rare instances. We thus consider the encoding cost of $\mathcal{X}^C$ as its anomalousness score; the higher the compression cost, the more likely it is "to arouse suspicion that it was generated by a different mechanism" [38].

$$norm = \arg\max_{r \in R} |\mathcal{S}_r^{-1}|,$$
$$score(\mathcal{X}^C) = < \mathcal{X}^C | \theta_{norm} >, \quad (12)$$

where $|\mathcal{S}_r^{-1}|$ is the total segment length of the regime $\theta_r$. Note that, in data streams, the concept of normal changes over time, and this is known as *concept drift* [21]. This approach can adaptively change the norm to judge incoming tensors as the concept drift.

LEMMA 4.1 (TIME COMPLEXITY OF CUBESCOPE). *The time complexity of CUBESCOPE is at least $O(N)$ and at most $O(N + R)$ per process, where $R$ is the number of regimes and $N$ is the total number of event entries in $\mathcal{X}^C$ (i.e., $\sum_{u_1} \cdots \sum_{u_M} \sum_t x_{u_1,...,u_M,t}$).*

PROOF. Please see Appendix B. □

## 5 EXPERIMENTS

In this section, we evaluate the performance of CUBESCOPE. We answer the following questions through the experiments.

- (Q1) *Effectiveness*: How successfully does CUBESCOPE discover compact description $C$ in given tensor streams?
- (Q2) *Accuracy*: How accurately does it achieve modeling, clustering, and streaming anomaly detection?
- (Q3) *Scalability*: How does it scale in terms of computational time?

**Datasets & Experimental Setup.** We use eight real datasets and four synthetics. The real datasets contain the event tensor streams

**Table 2: Dataset description**

| Dataset | The form of entry | Order |
|---|---|---|
| Local Mobility: Ride information attributes & timestamp → #rides | | |
| #1 NYC-Taxi [8] | (Pick-up/Drop-off location ID, Time) | 3 |
| #2 Bike-Share [2] | (User's age, Start/End station ID, Time) | 4 |
| E-commerce: Purchase information attributes & timestamp → #purchases | | |
| #3 Jewelry [4] | (Price, Brand, Gem, Accessory type, Time) | 4 |
| #4 Electronics [3] | (Brand, Item category, Time) | 3 |
| Network traffic/intrusion: Access detail attributes & timestamp → #accesses | | |
| #5 AirForce [5] | (Protocol type, Service, Flag, Land, Duration Src/Dst bytes, Wrong fragment, Urgent, Time) | 10 |
| #6 External [1] | (Proto, Src/Dst IP Addr, Src/Dst Pt, Flags,Duration,Packets,Bytes, Time) | 10 |
| #7 OpenStack [1] | " | 10 |
| #8 Kyoto [9] | (Src/Dst bytes, Count, Same srv/Serror/Srv serror rate, Dst host serror rate/same src port rate/srv serrors rate, Dst host count/srv count, Duration,Service,Flag,Time) | 15 |

of local mobility, e-commerce, and network traffic/intrusion and are summarized in Table 2. The synthetics and experimental settings are described in Appendix C.1.

**Baselines.** Our experiments are evaluated with twelve baselines.

Probabilistic generative models:
- Latent Dirichlet Allocation (LDA) [22] - A classical topic model, where the topic distribution is a multinomial.
- Neural Topic Model (NTM) [85] - A topic model based on neural variational inference.
- TriMine [68] - A factorization method for a high-order tensor, whose entries consists of multiple attributes and a timestamp.

Clustering approaches for time series, tensor, and data streams:
- K-means - The standard K-means clustering algorithm using Euclidean distance.
- TICC [37] - A clustering method for multivariate time series, where each cluster is characterized by a correlation network.
- CubeMarker [39] - An offline approach for discovering distinct patterns in tensor time series.
- Time-Aware LSTM (T-LSTM) [17] - A time series clustering method for sequences with irregular time intervals.
- DBSTREAM [36] - A clustering algorithm for evolving data streams, which incrementally updates the density of clusters.

Unsupervised anomaly detection methods:
- Local Outlier Factor (LOF) [24] - A density-based method for a collection of data points.
- Isolation forest (iForest) [56] - An offline method, where a forest of random cuts of data points isolates outliers.
- Robust Random Cut Forest (RRCF) [34] - A tree-based approach, which is designed for use with streaming data.
- MemStream [21] - A streaming approach using a denoising autoencoder and a memory module.

### 5.1 Q1.Effectiveness

We first demonstrate how effectively CUBESCOPE works on real datasets. Please also see the results in *Electronics* in Appendix C.2.

**Local Mobility.** The results for *NYC-Taxi* have already been presented in Figure 1. As already seen, CUBESCOPE identifies multiple regimes and their shifting points (Figure 1 (a)), and captures latent components (Figure 1 (b)(c)). These patterns reflect complicated social conditions and help us to understand human activities.

**Online Marketing Analytics.** Figure 3 shows our mining result for the *Jewelry* dataset. This dataset is an e-commerce-log collected

Figure 3: Market analysis of CUBESCOPE on the *Jewelry* dataset: (a) CUBESCOPE adaptively captured the changes in purchase behaviors caused by the sale. (b) The three components (Luxury, Middle, Affordable) for time. A darker color denotes a stronger relationship between each component and the time. It shows when the components attract consumer interest. (c) The three components for each attribute (price/brand/gem/accessory type) in Regime #1. Each of the columns shows four attributes. A darker color in the price rank and a larger size in the word cloud denote a stronger relationship with the component.



Figure 4: Real-time intrusion detection of CUBESCOPE on *AirForce* dataset: the stars indicate intrusions. It successfully identified the multiple types of intrusions of different durations (i.e., Regime #2: Smurf, Regime #3: Probe attacks, Regime #4: Neptune).

from an anonymous jewelry store. Each of the logs consists of four attributes, namely 20 prices, 6 anonymous brands, 32 gems, and 8 accessory types, with 12-hour timestamps. The price attribute is defined every fifty dollars up to 1K dollars i.e., 20 stages.

- *Regime identification:* As shown in Figure 3 (a), CUBESCOPE generates Regime #1 and starts monitoring the tensor stream. In late November, it detects a regime transition and generates Regime #2. Similarly, in late May, it generates a new Regime #3. These periods coincide with Black Friday [4] and Memorial Day [5]. This suggests our method captures the change in purchase behaviors caused by the sale.

- *Multi-aspect component analysis:* Figure 3 (b)(c) shows three components, which we manually named "Affordable", "Middle", and "Luxury". First, Figure 3 (b) shows the three components for the time attribute (i.e., **B**) in each regime. It demonstrates when each component attracts consumer interest. Figure 3 (c) shows the three components for each attribute (i.e., $\{\mathbf{A}^{(m)}\}_{m=1}^{4}$) in Regime #1. The components reveal the latent groups in four attributes (i.e., price, brand, gem, accessory type). It shows that each component (row) is strongly related to different brands or accessories.

Here, we provide the reader with some application scenarios. For targeted advertising and promotion strategies, analysts investigate purchase logs with millions, billions or even trillions [74] of events.

However, this approach requires expert knowledge and time resources. Since CUBESCOPE can automatically and efficiently summarize a massive amount of data into just a handful of components, it could provide analysts with a summary of the market or user preferences. Also, it is a more critical issue to analyze how the purchase behaviors change due to fads and sales. Our method recognizes the changes in dynamics as regime transitions and adaptively generates the summary for each regime.

**Cybersecurity.** We demonstrate the real-time intrusion detection of CUBESCOPE. Figure 4 shows the result for the *AirForce* dataset, which contains multiple intrusions simulated in a military network environment within 1.2 million records. We investigated the detected regimes and found that most corresponded to actual intrusions. For example, Regime #2 (orange) and Regime #4 (red) correspond to Smurf and Neptune attacks, respectively. Regime #3 (green) captures IP sweep/Stan/Port sweep. These intrusions are categorized as probe attacks [77]. Most importantly, these anomalies arise over time and thus their numbers, durations, and features are unknown in advance, whereas CUBESCOPE is fully automatic. It automatically recognizes anomalies and their types while updating the information for each type of anomaly in a streaming setting. We also conducted a quantitative analysis of this result in terms of the clustering and anomaly detection in Section 5.2.

## 5.2 Q2. Accuracy

We next evaluate the accuracy of CUBESCOPE in terms of modeling, clustering, and anomaly detection.

---

[4]Black Friday is a big sale event on the 4th Friday of November.
[5]On Memorial Day, most jewelry shops hold special sales.

Kota Nakamura, Yasuko Matsubara, Koki Kawabata, Yuhei Umeda, Yuichiro Wada, and Yasushi Sakurai
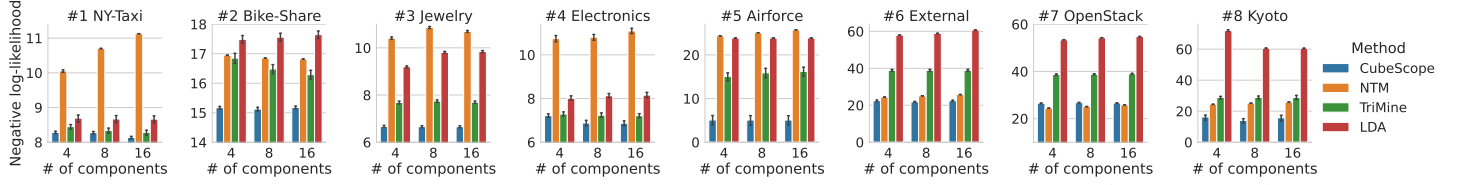


Figure 5: Modeling accuracy of CubeScope: the method consistently outperforms its baselines (lower is better).
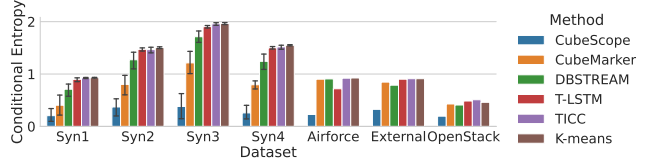


Figure 6: Clustering accuracy with respect to conditional entropy (lower is better).
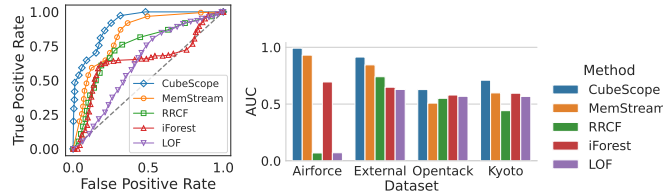


Figure 7: Detection accuracy (higher is better): CubeScope consistently wins. (left) The ROC curve on *External* dataset. (right) The ROC-AUC on all datasets.
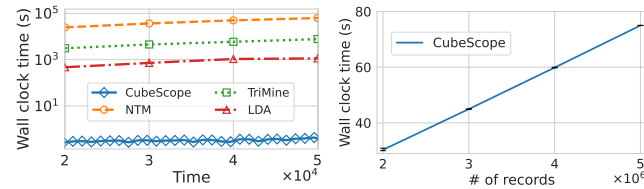


Figure 8: Scalability of CubeScope: (left) Wall clock time vs. data stream length. CubeScope surpasses its competitors at any time. It is up to 312,000x faster than the baselines. (right) Average wall clock time vs. # of records in a process. The algorithm scales linearly.

**Modeling.** We compared the modeling accuracy of CubeScope and the probabilistic generative models. Figure 5 shows the average negative log-likelihood of every current tensor $\mathcal{X}^C$ of length 168 for each model. For a fair comparison, we use 4, 8, and 16 components/topics for all models. A lower value indicates better model construction. Unsurprisingly, CubeScope achieves high modeling accuracy on all datasets because it can capture high-order tensor streams. Since LDA and NTM handle a tensor as a large matrix, they cannot capture multi-aspect features. TriMine is designed for a high-dimensional and sparse tensor, but it cannot capture tensor streams containing various time-evolving patterns (i.e., regimes).

**Clustering.** Next, we show how accurately CubeScope can find regimes. We use both labeled datasets and synthetics because it is insufficient to evaluate only real datasets containing some time-evolving patterns that are not repeated (i.e., a few clusters appear once). We generated four types of synthetics as follows [37] (see Appendix C.1 for details), and evaluated them ten times and reported the mean and standard deviation values. Finally, we compare a standard measure of conditional entropy (CE) from the confusion matrix (CM) of the prediction regime labels against true cluster

labels. The CE score shows the difference between two clusters using the following equation: $CE = -\Sigma_{i,j} \frac{CM_{i,j}}{\Sigma_{i,j}CM_{i,j}} \log \frac{CM_{i,j}}{\Sigma_j CM_{i,j}}$. Note that an ideal confusion matrix must be diagonal, in which case $CE = 0$. Figure 6 compares CubeScope with clustering methods. Our method consistently outperforms its competitors because it can handle high-dimensional and sparse tensors. TICC failed to capture sparse sequences. T-LSTM is designed for sequences with sparsity but cannot handle high-dimensional tensors. CubeMarker can capture tensor time series but cannot handle sparse tensors. DBSTREAM has the ability to recognize clusters in data streams but cannot capture multi-aspect features in tensor streams.

**Anomaly Detection.** We evaluate anomaly detection performance for four real datasets containing ground truth anomalies. We first compute anomaly scores for CubeScope and unsupervised baselines, and then select the top-$k$ most anomalous periods ($k = 20, 40, \ldots$). Next, we compute true and false positive rates for each method's output. Figure 7 shows the ROC curve for *External* dataset and ROC-AUC for all datasets. A higher value indicates better detection accuracy. CubeScope achieves a high detection accuracy for every dataset, while other methods cannot detect anomalies very well because only our approach captures dynamical multi-aspect patterns and utilizes them for subsequent anomaly detection.

### 5.3 Q3. Scalability

Finally, we evaluate the computational time needed by CubeScope for large tensor streams. The left part of Figure 8 shows the wall clock time of an experiment performed on a large *NYC-Taxi* dataset. Thanks to the incremental update, our method is independent of data stream length. In fact, our method achieved a constant computation time, which was up to five orders of magnitude faster than its baselines. The right part of Figure 8 shows the computational time of C-Decomposer when varying the size of an input tensor. Since CubeScope achieves fast and efficient model estimation for $O(N)$ time (as discussed in Lemma 4.1), the complexity scales linearly with respect to the number of events.

## 6 CONCLUSION

In this paper, we focused on the dynamic summarization of high-order event tensor streams and presented CubeScope, which exhibits all the desirable properties that we listed in the introduction;

- *Effective*: it incrementally captures dynamical multi-aspect patterns and summarizes a semi-infinite collection of event tensor streams into an interpretable representation.
- *General*: our experiments with various datasets showed that CubeScope successfully discovers meaningful patterns and anomalies, and outperforms state-of-the-art modeling, clustering, and anomaly detection methods.
- *Scalable*: its computational time is constant and independent of the input data length and the dimensionality in each attribute.

## ACKNOWLEDGMENTS

## REFERENCES

[1] [n.d.]. *CIDDS (Coburg Intrusion Detection Data Sets)*. https://www.hs-coburg.de/forschung/forschungsprojekte-oeffentlich/informationstechnologie/cidds-coburg-intrusion-detection-data-sets.html.
[2] [n.d.]. *Citi Bike Trip Histories*. https://ride.citibikenyc.com/system-data.
[3] [n.d.]. *eCommerce purchase history from electronics store*. https://www.kaggle.com/mkechinov/ecommerce-purchase-history-from-electronics-store.
[4] [n.d.]. *eCommerce purchase history from jewelry store*. https://www.kaggle.com/mkechinov/ecommerce-purchase-history-from-jewelry-store.
[5] [n.d.]. *KDD Cup 1999 Data*. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.
[6] [n.d.]. *CubeScope*. https://github.com/kotaNakm/CubeScope.
[7] [n.d.]. *River:online machine learning in Python*. https://riverml.xyz/dev/api/cluster/DBSTREAM/.
[8] [n.d.]. *TLC Trip Record Data*. https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page.
[9] [n.d.]. *Traffic Data from Kyoto University's Honeypots*. https://www.takakura.com/Kyoto_data/.
[10] 2019. Predicting pregnancy using large-scale datafrom a women's health tracking mobile application. In *WWW*. 2999–3005.
[11] 2020. https://www1.nyc.gov/assets/home/downloads/pdf/executive-orders/2020/eeo-100.pdf.
[12] 2020. https://www.state.gov/wp-content/uploads/2020/03/2020-03-20-Notice-New-York-on-Pause-Order.pdf.
[13] 2020. https://www.governor.ny.gov/news/governor-cuomo-announces-new-york-city-enter-phase-1-reopening-june-8-and-five-regions-enter.
[14] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. 2003. A Framework for Clustering Evolving Data Streams. In *VLDB*. 81–92.
[15] Leman Akoglu, Hanghang Tong, and Danai Koutra. 2015. Graph based anomaly detection and description: a survey. *Data mining and knowledge discovery* 29, 3 (2015), 626–688.
[16] Miguel Araujo, Spiros Papadimitriou, Stephan Günnemann, Christos Faloutsos, Prithwish Basu, Ananthram Swami, Evangelos E Papalexakis, and Danai Koutra. 2014. Com2: fast automatic discovery of temporal ('comet') communities. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 271–283.
[17] Inci M. Baytas, Cao Xiao, Xi Zhang, Fei Wang, Anil K. Jain, and Jiayu Zhou. 2017. Patient Subtyping via Time-Aware LSTM Networks. In *KDD*. 65–74.
[18] Alex Beutel, Kenton Murray, Christos Faloutsos, and Alexander J Smola. 2014. Cobafi: collaborative bayesian filtering. In *WWW*. 97–108.
[19] Siddharth Bhatia, Arjit Jain, Pan Li, Ritesh Kumar, and Bryan Hooi. 2021. MStream: Fast Anomaly Detection in Multi-Aspect Streams. In *WWW*. ACM / IW3C2, 3371–3382.
[20] Siddharth Bhatia, Arjit Jain, Pan Li, Ritesh Kumar, and Bryan Hooi. 2021. MStream: Fast Anomaly Detection in Multi-Aspect Streams. In *WWW*. 3371–3382.
[21] Siddharth Bhatia, Arjit Jain, Shivin Srivastava, Kenji Kawaguchi, and Bryan Hooi. 2022. MemStream: Memory-Based Streaming Anomaly Detection. In *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, Frédérique Laforest, Raphaël Troncy, Elena Simperl, Deepak Agarwal, Aristides Gionis, Ivan Herman, and Lionel Médini (Eds.). ACM, 610–621.
[22] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research* 3 (2003), 993–1022.
[23] Christian Böhm, Christos Faloutsos, Jia-Yu Pan, and Claudia Plant. 2007. Ric: Parameter-free noise-robust clustering. *TKDD* 1, 3 (2007), 10–es.
[24] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. LOF: Identifying Density-Based Local Outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD '00)*. Association for Computing Machinery, New York, NY, USA, 93–104.
[25] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly Detection: A Survey. *ACM Comput. Surv.* 41, 3, Article 15 (jul 2009), 58 pages.
[26] Pudi Chen, Shenghua Liu, Chuan Shi, Bryan Hooi, Bai Wang, and Xueqi Cheng. 2018. NeuCast: Seasonal Neural Forecast of Power Grid Time Series.. In *IJCAI*. 3315–3321.
[27] Lianhua Chi, Bin Li, Xingquan Zhu, Shirui Pan, and Ling Chen. 2017. Hashing for adaptive real-time graph stream classification with concept drifts. *IEEE transactions on cybernetics* 48, 5 (2017), 1591–1604.
[28] Eunjoon Cho, Seth A. Myers, and Jure Leskovec. 2011. Friendship and mobility: user movement in location-based social networks. In *KDD*. 1082–1090.
[29] Gianmarco De Francisci Morales, Albert Bifet, Latifur Khan, Joao Gama, and Wei Fan. 2016. Iot big data stream mining. In *KDD*. 2119–2120.
[30] Shohreh Deldari, Daniel V. Smith, Hao Xue, and Flora D. Salim. 2021. Time Series Change Point Detection with Self-Supervised Contrastive Predictive Coding. In *WWW*. ACM / IW3C2, 3124–3135.
[31] Hadi Fanaee-T and João Gama. 2016. Tensor-based anomaly detection: An interdisciplinary survey. *Knowledge-Based Systems* 98 (2016), 130–147.
[32] Shufeng Gong, Yanfeng Zhang, and Ge Yu. 2017. Clustering stream data by exploring the evolution of density mountain. *Proceedings of the VLDB Endowment* 11, 4 (2017), 393–405.
[33] Peter D Grünwald, In Jae Myung, and Mark A Pitt. 2005. *Advances in minimum description length: Theory and applications*. MIT press.
[34] Sudipto Guha, Nina Mishra, Gourav Roy, and Okke Schrijvers. 2016. Robust Random Cut Forest Based Anomaly Detection on Streams. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48 (ICML'16)*. JMLR.org, 2712–2721.
[35] Manish Gupta, Jing Gao, Charu C Aggarwal, and Jiawei Han. 2013. Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and data Engineering* 26, 9 (2013), 2250–2267.
[36] Michael Hahsler and Matthew Bolaños. 2016. Clustering data streams based on shared density between micro-clusters. *IEEE Transactions on Knowledge and Data Engineering* 28, 6 (2016), 1449–1461.
[37] David Hallac, Sagar Vare, Stephen Boyd, and Jure Leskovec. 2017. Toeplitz inverse covariance-based clustering of multivariate time series data. In *KDD*.
[38] Douglas M Hawkins. 1980. *Identification of outliers*. Vol. 11. Springer.
[39] Takato Honda, Yasuko Matsubara, Ryo Neyama, Mutsumi Abe, and Yasushi Sakurai. 2019. Multi-aspect mining of complex sensor sequences. In *ICDM*.
[40] Bryan Hooi, Shenghua Liu, Asim Smailagic, and Christos Faloutsos. 2017. BeatLex: Summarizing and Forecasting Time Series with Patterns. In *PKDD*, Vol. 10535. 3–19.
[41] Bryan Hooi, Kijung Shin, Shenghua Liu, and Christos Faloutsos. 2019. SMF: Drift-aware matrix factorization with seasonal patterns. In *SIAM*. 621–629.
[42] Wenjie Hu, Yang Yang, Ziqiang Cheng, Carl Yang, and Xiang Ren. 2021. Time-series event prediction with evolutionary state graph. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*.
[43] Tomoharu Iwata, Shinji Watanabe, Takeshi Yamada, and Naonori Ueda. 2009. Topic tracking model for analyzing consumer purchase behavior. In *IJCAI*.
[44] Jun-Gi Jang and U Kang. 2021. Fast and Memory-Efficient Tucker Decomposition for Answering Diverse Time Range Queries. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 725–735.
[45] Meng Jiang, Alex Beutel, Peng Cui, Bryan Hooi, Shiqiang Yang, and Christos Faloutsos. 2015. A general suspiciousness metric for dense blocks in multimodal data. In *2015 IEEE International Conference on Data Mining*. IEEE, 781–786.
[46] Harshavardhan Kamarthi, Lingkai Kong, Alexander Rodríguez, Chao Zhang, and B Aditya Prakash. 2022. CAMul: Calibrated and Accurate Multi-view Time-Series Forecasting. In *Proceedings of the ACM Web Conference 2022*. 3174–3185.
[47] Koki Kawabata, Yasuko Matsubara, Takato Honda, and Yasushi Sakurai. 2020. Non-Linear Mining of Social Activities in Tensor Streams. In *KDD*. 2093–2102.
[48] Koki Kawabata, Yasuko Matsubara, and Yasushi Sakurai. 2019. Automatic sequential pattern mining in data streams. In *CIKM*. 1733–1742.
[49] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).
[50] Tamara G Kolda and Brett W Bader. 2009. Tensor decompositions and applications. *SIAM review* 51, 3 (2009), 455–500.
[51] Danai Koutra, U Kang, Jilles Vreeken, and Christos Faloutsos. 2014. Vog: Summarizing and understanding large graphs. In *Proceedings of the 2014 SIAM international conference on data mining*. SIAM, 91–99.
[52] Mathias Kraus and Stefan Feuerriegel. 2019. Personalized purchase prediction of market baskets with Wasserstein-based sequence matching. In *KDD*. 2643–2652.
[53] Changhee Lee and Mihaela Van Der Schaar. 2020. Temporal phenotyping using deep predictive clustering of disease progression. In *ICML*. 5767–5777.
[54] Lei Li, B Aditya Prakash, and Christos Faloutsos. 2010. *Parsimonious linear fingerprinting for time series*. Technical Report. CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE.
[55] Xiangsheng Li, Jiaxin Mao, Weizhi Ma, Yiqun Liu, Min Zhang, Shaoping Ma, Zhaowei Wang, and Xiuqiang He. 2021. Topic-Enhanced Knowledge-Aware Retrieval Model for Diverse Relevance Estimation. In *WWW*. 756–767.
[56] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2012. Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 6, 1 (2012), 1–39.
[57] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. 2018. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering* 31, 12 (2018), 2346–2363.
[58] Yue Lu, Renjie Wu, Abdullah Mueen, Maria A Zuluaga, and Eamonn Keogh. 2022. Matrix Profile XXIV: Scaling Time Series Anomaly Detection to Trillions of Datapoints and Ultra-fast Arriving Data Streams. In *Proceedings of the 28th*

*ACM SIGKDD Conference on Knowledge Discovery and Data Mining.* 1173–1182.

[59] Qianli Ma, Jiawei Zheng, Sen Li, and Gary W Cottrell. 2019. Learning representations for time series clustering. *Advances in neural information processing systems* 32 (2019), 3781–3791.

[60] Stratos Mansalis, Eirini Ntoutsi, Nikos Pelekis, and Yannis Theodoridis. 2018. An evaluation of data stream clustering algorithms. *Statistical Analysis and Data Mining: The ASA Data Science Journal* 11, 4 (2018), 167–187.

[61] Emaad Manzoor, Hemank Lamba, and Leman Akoglu. 2018. xstream: Outlier detection in feature-evolving data streams. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 1963–1972.

[62] Hing-Hao Mao, Chung-Jung Wu, Evangelos E Papalexakis, Christos Faloutsos, Kuo-Chen Lee, and Tien-Cheu Kao. 2014. Malspot: Multi 2 malicious network behavior patterns analysis. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining.* Springer, 1–14.

[63] Yasuko Matsubara and Yasushi Sakurai. 2016. Regime Shifts in Streams: Real-time Forecasting of Co-evolving Time Sequences. In *KDD.* 1045–1054.

[64] Yasuko Matsubara and Yasushi Sakurai. 2019. Dynamic Modeling and Forecasting of Time-Evolving Data Streams. In *KDD.* 458–468.

[65] Yasuko Matsubara, Yasushi Sakurai, and Christos Faloutsos. 2014. AutoPlait: Automatic Mining of Co-evolving Time Sequences. In *SIGMOD.*

[66] Yasuko Matsubara, Yasushi Sakurai, and Christos Faloutsos. 2015. The Web as a Jungle: Non-Linear Dynamical Systems for Co-evolving Online Activities. In *WWW.*

[67] Yasuko Matsubara, Yasushi Sakurai, and Christos Faloutsos. 2016. Non-Linear Mining of Competing Local Activities. In *WWW.*

[68] Yasuko Matsubara, Yasushi Sakurai, Christos Faloutsos, Tomoharu Iwata, and Masatoshi Yoshikawa. 2012. Fast mining and forecasting of complex time-stamped events. In *KDD.* 271–279.

[69] Charalampos Mavroforakis, Isabel Valera, and Manuel Gomez-Rodriguez. 2017. Modeling the Dynamics of Learning Activity on the Web. In *WWW.* ACM, 1421–1430.

[70] Yu Meng, Yunyi Zhang, Jiaxin Huang, Yu Zhang, Chao Zhang, and Jiawei Han. 2020. Hierarchical Topic Mining via Joint Spherical Tree and Text Embedding. In *KDD.* 1908–1917.

[71] Maya Okawa, Tomoharu Iwata, Takeshi Kurashima, Yusuke Tanaka, Hiroyuki Toda, and Naonori Ueda. 2019. Deep Mixture Point Processes: Spatio-temporal Event Prediction with Rich Contextual Information. In *KDD.* 373–383.

[72] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12 (2011), 2825–2830.

[73] Ian Porteous, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling. 2008. Fast collapsed gibbs sampling for latent dirichlet allocation. In *KDD.* 569–577.

[74] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. 2012. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining.* 262–270.

[75] Yasushi Sakurai, Yasuko Matsubara, and Christos Faloutsos. 2016. Mining Big Time-series Data on the Web. In *WWW.* 1029–1032.

[76] Aaron Schein, John Paisley, David M Blei, and Hanna Wallach. 2015. Bayesian poisson tensor factorization for inferring multilateral relations from sparse dyadic event counts. In *KDD.* 1045–1054.

[77] Benedetto Marco Serinelli, Anastasija Collen, and Niels Alexander Nijdam. 2020. Training guidance with kdd cup 1999 and nsl-kdd data sets of anidinr: Anomaly-based network intrusion detection system. *Procedia Computer Science* 175 (2020), 560–565.

[78] Neil Shah, Danai Koutra, Tianmin Zou, Brian Gallagher, and Christos Faloutsos. 2015. TimeCrunch: Interpretable Dynamic Graph Summarization. In *KDD.*

[79] Lei Shi, Aryya Gangopadhyay, and Vandana P Janeja. 2015. STenSr: Spatio-temporal tensor streams for anomaly detection and pattern discovery. *Knowledge and Information Systems* 43, 2 (2015), 333–353.

[80] Kijung Shin, Bryan Hooi, Jisu Kim, and Christos Faloutsos. 2017. Densealert: Incremental dense-subtensor detection in tensor streams. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* 1057–1066.

[81] Koen Smets and Jilles Vreeken. 2011. The odd one out: Identifying and characterising anomalies. In *Proceedings of the 2011 SIAM international conference on data mining.* SIAM, 804–815.

[82] Tsubasa Takahashi, Bryan Hooi, and Christos Faloutsos. 2017. Autocyclone: Automatic mining of cyclic online activities with robust tensor factorization. In *WWW.* 213–221.

[83] Veronica Tozzo, Federico Ciech, Davide Garbarino, and Alessandro Verri. 2021. Statistical Models Coupling Allows for Complex Local Multivariate Time Series Analysis. In *KDD.* 1593–1603.

[84] Yichen Wang, Robert Chen, Joydeep Ghosh, Joshua C Denny, Abel Kho, You Chen, Bradley A Malin, and Jimeng Sun. 2015. Rubik: Knowledge guided tensor factorization and completion for health data analytics. In *KDD.* 1265–1274.

[85] Yue Wang, Jing Li, Hou Pong Chan, Irwin King, Michael R. Lyu, and Shuming Shi. 2019. Topic-Aware Neural Keyphrase Generation for Social Media Language. In *ACL.* 2516–2526.

[86] Yuan Xue, Denny Zhou, Nan Du, Andrew M. Dai, Zhen Xu, Kun Zhang, and Claire Cui. 2020. Deep State-Space Generative Model For Correlated Time-to-Event Predictions. In *KDD.* ACM, 1552–1562.

[87] Jaewon Yang, Julian McAuley, Jure Leskovec, Paea LePendu, and Nigam Shah. 2014. Finding progression stages in time-evolving event sequences. In *WWW.* 783–794.

[88] Shuochao Yao, Shaohan Hu, Yiran Zhao, Aston Zhang, and Tarek F. Abdelzaher. 2017. DeepSense: A Unified Deep Learning Framework for Time-Series Mobile Sensing Data Processing. In *WWW.* ACM, 351–360.

[89] Jianhua Yin, Daren Chao, Zhongkun Liu, Wei Zhang, Xiaohui Yu, and Jianyong Wang. 2018. Model-based clustering of short text streams. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining.* 2634–2642.

**Table 3: Symbols and definitions.**

| Symbol | Definition |
|---|---|
| $M$ | Number of attributes of complex event tensor |
| $U_1 \dots U_M$ | A set consisting of a number of unique units in each attribute |
| $T$ | Length of whole tensor stream |
| $\tau$ | Length of current tensor |
| $\mathcal{X}$ | Whole event tensor stream, $\mathcal{X} \in \mathbb{N}^{U_1 \times \cdots \times U_M \times T}$ |
| $\mathcal{X}^C$ | Current event tensor, $\mathcal{X}^C \in \mathbb{N}^{U_1 \times \cdots \times U_M \times \tau}$ |
| $K$ | Number of latent components |
| $\mathbf{A}^{(m)}$ | $m$-th attribute component matrix, $K \times U_m$ |
| $\mathbf{B}$ | Time component matrix, $\tau \times K$ |
| $Q$ | FIFO queue for retaining past component matrices |
| $L$ | The size of queue $Q$ |
| $R$ | Number of regimes |
| $\theta_r$ | $r$-th regime parameter, i.e., $\theta_r = \{\mathbf{A}^{(1)}, \dots, \mathbf{A}^{(M)}, \mathbf{B}\}$ |
| $\Theta$ | Regime set, i.e., $\Theta = \{\theta_1, \dots, \theta_R\}$ |
| $G$ | Number of regime assignments (i.e., segments) |
| $s_g$ | Trajectory of shift to $r$-th regime at time $t_s$, i.e., $s_g = (t_s, r)$ |
| $\mathcal{S}$ | Regime assignments, i.e., $\mathcal{S} = \{s_1, \dots, s_G\}$ |
| $|\mathcal{S}_r^{-1}|$ | Total segment length of the regime $\theta_r$ |
| $C$ | Compact description, i.e., $C = \{R, \Theta, G, \mathcal{S}\}$ |
| $score(\mathcal{X}^C)$ | Anomalousness score of $\mathcal{X}^C$ |

---

**Algorithm 1** CubeScope $(\mathcal{X}^C, C, Q)$

---

**Input:** 1. Current tensor $\mathcal{X}^C \in \mathbb{N}^{U_1 \times \dots \times U_M \times \tau}$
     2. Previous candidate solution $C = \{R, \Theta, G, \mathcal{S}\}$
     3. Previous past parameter set $Q$
**Output:** 1. Updated candidate solution $C'$
     2. Updated past parameter set $Q'$
     3. Anomalousness score $score(\mathcal{X}^C)$
1: $\theta_c, Q'$ = C-Decomposer $(\mathcal{X}^C, Q)$;
2: $C', score(\mathcal{X}^C)$ = C-Compressor $(\theta_c, \mathcal{X}^C, C)$;
3: **return** $C', Q', score(\mathcal{X}^C)$;

---

**Algorithm 2** C-Decomposer $(\mathcal{X}^C, Q)$

---

**Input:** 1. Current tensor $\mathcal{X}^C \in \mathbb{N}^{U_1 \times \dots \times U_M \times \tau}$
     2. Previous past parameter set $Q$
**Output:** 1. Current model parameter set $\theta_c = \{\mathbf{A}^{(1)}, \dots \mathbf{A}^{(M)}, \mathbf{B}\}$
     2. Updated past parameter set $Q'$
1: **for each** iteration **do**
2:   **for each** non-zero element $x$ in $\mathcal{X}^C$ **do**
3:     **for each** entry for $x$ **do**
4:       Draw hidden variable $z$; // According to Eq. (7)
5:     **end for**
6:   **end for**
7: **end for**
8: Compute $\mathbf{A}^{(1)}, \dots, \mathbf{A}^{(M)}, \mathbf{B}$; //According to Eq. (8)
9: $\theta_c \leftarrow \mathbf{A}^{(1)}, \dots, \mathbf{A}^{(M)}, \mathbf{B}$;
10: $Q$.deque; // Remove the oldest set of component matrices
11: $Q' \leftarrow Q$.enque$(\theta_c)$; // Insert a set of current estimated matrices $\theta_c$
12: **return** $\theta_c, Q'$;

# APPENDIX

# A PROPOSED MODEL

Table 3 lists the symbols and definitions used in this paper. All logarithms are to base 2, and by convention we use $0 \log 0 = 0$.

# B STREAMING ALGORITHM

Algorithm 1 shows the overall procedure for CubeScope, which composed of C-Decomposer (Algorithm 2) and C-Compressor (Algorithm. 3). C-Decomposer continuously monitors a current tensor $\mathcal{X}^C$ and generates a regime $\theta_c$. Then, C-Compressor updates the compact description $C$ with $\theta_c$ and measures the anomalousness of $\mathcal{X}^C$.

---

**Algorithm 3** C-Compressor $(\theta_c, \mathcal{X}^C, C)$

---

**Input:** 1. Candidate model parameter set $\theta_c = \{\mathbf{A}^{(1)}, \dots \mathbf{A}^{(M)}, \mathbf{B}\}$
     2. New observation tensor $\mathcal{X}^C \in \mathbb{N}^{U_1 \times \dots \times U_M \times \tau}$
     3. Previous compact description $C = \{R, \Theta, G, \mathcal{S}\}$
**Output:** 1. Updated compact description $C' = \{R', \Theta', G', \mathcal{S}'\}$
     2. Anomalousness score $score(\mathcal{X}^C)$
1: /* Update compact description $C$ */
2: **if** $< \mathcal{X}^C; \theta_p >$ is less than $< \mathcal{X}^C; \theta_c >$ **then**
3:   /* Stay in the previous regime $\theta_p$ */
4:   $\theta_p' \leftarrow$ Regime update$(\theta_p, \theta_c)$; // According to Eq. (11)
5: **else**
6:   $\theta_e = \arg\min_{\theta \in \Theta} < \mathcal{X}^C; \theta >$;
7:   **if** $< \mathcal{X}^C; \theta_c >$ is less than $< \mathcal{X}^C; \theta_e >$ **then**
8:     /* Shift to the candidate regime $\theta_c$ */
9:     $R' \leftarrow R + 1; \Theta' \leftarrow \Theta \cup \theta_c$;
10:     $G' \leftarrow G + 1; \mathcal{S}' \leftarrow \mathcal{S} \cup (t, R+1)$;
11:   **else**
12:     /* Shift to the existing regime $\theta_e$ */
13:     $\theta_e' \leftarrow$ Regime update$(\theta_e, \theta_c)$; // According to Eq. (11)
14:     $G' \leftarrow G + 1; \mathcal{S}' \leftarrow \mathcal{S} \cup (t, e)$;
15:   **end if**
16: **end if**
17: /* Compute anomalousness score*/
18: $norm \leftarrow \arg\max_{r \in R} |\mathcal{S}_r^{-1}|$;
19: $score(\mathcal{X}^C) \leftarrow < \mathcal{X}^C | \theta_{norm} >$;
20: **return** $C' = \{R', \Theta' G' \mathcal{S}'\}, score(\mathcal{X}^C)$;
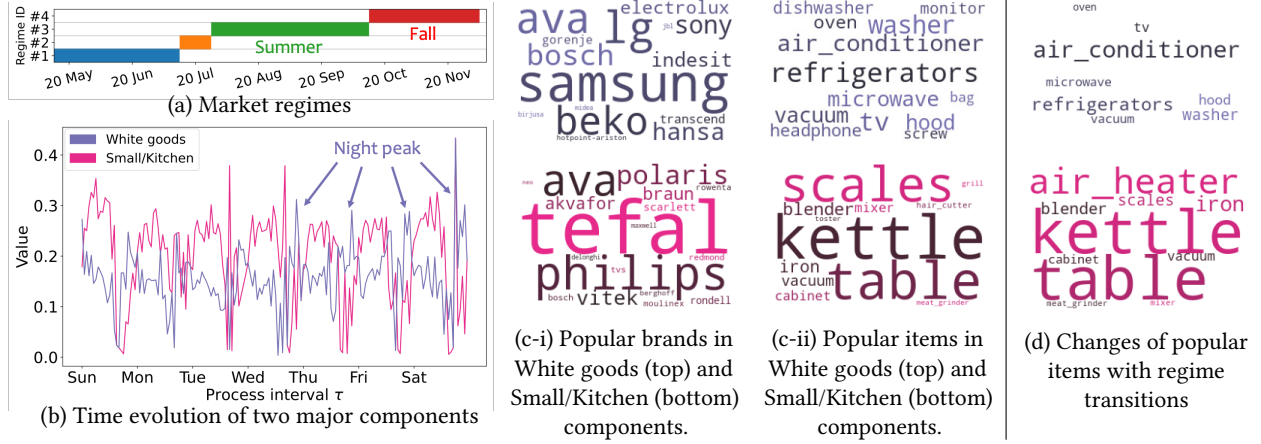
**Proof of Lemma 4.1.**

Proof. For each time point, CubeScope first runs C-Decomposer, which draws hidden variables $z_{u,t}$ with each entry for non-zero element $x_{u,t}$ in $\mathcal{X}^C$. This process requires $O(\#iter \cdot KN)$, where $\#iter$ is the number of iterations for drawing $z$, $K$ is the number of components, and $N$ is the total number of event entries in $\mathcal{X}^C$ (i.e., $\sum_{u_1} \cdots \sum_{u_M} \sum_t x_{u_1, \dots, u_M, t}$). Since $\#iter$ and $K$ are small values and constant, they are negligible. Thus, the complexity of C-Decomposer is $O(N)$. In C-Compressor, it tracks $\theta_c$ and $\theta_p$. If it employs the previous regime $\theta_p$ for current tensor $\mathcal{X}^C$, it can quickly update the regime, which requires $O(1)$ time. Otherwise, it then tries to find the optimal regime in $\Theta$, which requires $O(R)$ time. Overall, CubeScope needs these two algorithms. Thus, the complexity is at least $O(N)$ and at most $O(N + R)$ per process. □

**Model Initialization.** When we start the iteration with regime set $\Theta$ and $Q$ for past parameter set, we uniformly take several sample segments with interval $\tau$ from initial tensor $\mathcal{X}_s$ and then estimate the model parameters $\theta_s$ for each. The most appropriate regime set is determined by monitoring the total encoding cost while increasing the number of model parameters $\theta_s$:

$$\Theta = \arg\min_{\Theta \in \Theta_s} < \mathcal{X}_s; \Theta >, \tag{13}$$

where $\Theta_s = \{\theta_{s1}, \theta_{s2}, \dots\}$ is a set of regimes estimated from each sample. We also set $L$ to the average interval of the shifting points (i.e., the average length of segments).

**Process Interval $\tau$.** The parameter $\tau$ determines the size of a current tensor, as well as the minimum granularity of regimes. Users need to know regimes under various granularities (e.g., daily and weekly patterns), thus $\tau$ is generally chosen depending on the application. The runtime of C-Decomposer scales linearly with the number of records in a current tensor rather than the size of $\tau$. Although a larger $\tau$ imposes the algorithm to process a larger current tensor, there should be a small impact on the runtime because we assume sparse tensor streams.

(a) Market regimes

(b) Time evolution of two major components

(c-i) Popular brands in White goods (top) and Small/Kitchen (bottom) components.

(c-ii) Popular items in White goods (top) and Small/Kitchen (bottom) components.

(d) Changes of popular items with regime transitions

**Figure 9: Customer behavior modeling on *Electronics*: (a) CUBESCOPE discovered a total of four regimes that reflect the seasonality of customer behavior. (b) Two components show clear daily periodicity; The White goods component spikes at night. The Small/Kitchen component shows high peaks during the daytime. (c) Popular brand/item categories for each component. A larger size denotes a closer relationship. (d) Popular item categories change with regime transitions. Top: air conditioners become the most frequent item in Regime #2 (early Summer). Bottom: air heaters are the popular items in Regime #4 (Fall).**

## C   EXPERIMENTS

### C.1   Experimental Setup

We conducted our experiments on an Intel Xeon E5-2637 3.5GHz quad core CPU with 192GB of memory and running Linux.

**Generating the Datasets.** We first generate four types of sparse event tensors (1, 2, 3, and 4), which have $100K$ observations as $\mathcal{X} \in \mathbb{N}^{100\times100\times100\times100}$. Each attribute of an event entry is drawn from multinomial distributions whose parameter is defined by random values $[0.1, 0.5]$ and a Dirichlet prior. Finally, four different synthetic datasets are built by using different combinations of event tensors as follows [37]: "1,2,1", "1,2,3,2,1", "1,2,3,4,1,2,3,4", "1,2,2,1,3,3,3,1".

**Implementaion & Parameters.** We used the open-source implementation of LDA, K-means, LOF, and iForest in [72]. For NTM, we implemented it based on the pytorch framework and applied Adam optimization with a learning rate of $1e-3$, following the design and the parameter setting in [85]. We also used open-sourced implementations of TICC [37], T-LSTM [17], RRCF [34], CubeMarker [39], and MemStream [21], provided by the authors, following parameter settings as suggested in the original papers. For a fair comparison in terms of computational time, we implemented TriMine in Python, following C implementation provided by the authors. The input for LDA/NTM is bag-of-words representations of all the categories, i.e., $\mathbf{W} \in \mathbb{N}^{\tau \times (U_1 + \cdots + U_M)}$. In evaluation of clustering accuracy, the width of a current tensor is set with 10. Since TICC and T-LSTM need to specify the number of clusters, we set the true number of clusters. DBSTREAM, which is implemented in [7], and CUBESCOPE are automatically determine the number of clusters. We set the radius of each micro-clusters as 8.5 for DBSTREAM, and the number of components $K = 8$ for CUBESCOPE. To validate detection accuracy, we set $\tau = 1$ for all methods. We used a $5\tau$ length of the stream to conduct the model initialization for CUBESCOPE.

### C.2   Effectiveness

We also demonstrate how effectively CUBESCOPE works on the *Electronics* dataset.

**Online Marketing Analytics.** Figure 9 shows stream mining results of CUBESCOPE on the *Electronics*. This data is the purchase data obtained over a year from a large home appliances and electronics online store. The data contains a list of two attributes; 867 brands and 124 item categories with an hourly timestamp.

- *Regime identification:* CUBESCOPE discovered four type of regimes in Figure 9 (a). Specifically, our method found Regime #2 during a short period around July, and then discovered Regime #3 for the summer season and Regime #4 for the fall season. This result shows that the behaviors of purchases shift with the transition of seasons.

- *Multi-aspect component analysis:* Figure 9 (b) shows the temporal evolution of two major components, which are shown in the time component matrix $\mathbf{B}$ in Regime #1. We manually named the two components "White goods" and "Small/Kitchen". These components exhibit contrasting behavior. The White goods sequence peaks at night, while the Small/Kitchen sequence peaks during the daytime. Figure 9 (c) shows the attribute component matrices $\{\mathbf{A}^{(m)}\}_{m=1}^{2}$ in Regime #1, namely the latent relationships between two components (row) and two attributes (column). Figure 9 (d) shows the changes of popular item categories in association with regime transitions. These changes make sense. As shown in the top figure, the White goods component in Regime #2 (early summer) has the strongest relationship with an air conditioner. Similarly, the bottom figure shows the Small/Kitchen component in Regime #4 (Fall), where an air heater appeared as a popular item.