LIVE-ITS: LSH-based Interactive Visualization Explorer for Large-Scale Incomplete Time Series

Hongjie Chen
Dolby Labs.
Atlanta, USA
hongjie.chen@dolby.com

Aaron D. Beachnau
Dolby Labs.
Atlanta, USA
ADB@dolby.com

Panos Thomas
Dolby Labs.
Atlanta, USA
panos.thomas@dolby.com

Pranav Maneriker
Dolby Labs.
Atlanta, USA
pranav.maneriker@dolby.com

Josh Kimball
Dolby Labs.
Atlanta, USA
josh.kimball@dolby.com

Ryan A. Rossi
Adobe Research
San Jose, USA
ryrossi@adobe.com

2024 IEEE International Conference on Big Data (BigData) | 979-8-3503-6248-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/BigData62323.2024.10825096

***Abstract*—Recent advances in time series research have created a significant demand for better time series visualization techniques, especially for large-scale datasets that contain millions of time series or more. In this paper, we consider a common use scenario where analysts aim to identify representative time series from a large collection. These representative time series generalize as many time series as possible and can be used for downstream tasks. Building a visualization system for this scenario involves many challenges, including visualizing, selecting, and highlighting a subset of time series from the overall dataset. Moreover, the potential for time series to be incomplete due to missing records adds an extra layer of difficulty. To address these challenges, we propose a novel visualization system, called the Locality Sensitive Hashing-based Interactive Visualization Explorer for large-scale Incomplete Time Series (LIVE-ITS). On the frontend, LIVE-ITS allows analysts to interact with the system and select representative time series in areas of interest. In the backend, LIVE-ITS not only selects an optimal subset that represents as many time series as possible, but also achieves the best possible time complexity. Experiments on both synthetic dataset and real-world datasets show that LIVE-ITS exhibits high partition accuracy and high response efficiency, further validating the effectiveness of our proposed visualization system.**

***Index Terms*—large-scale time-series, incomplete time-series, time-series visualization, visualization analytics**

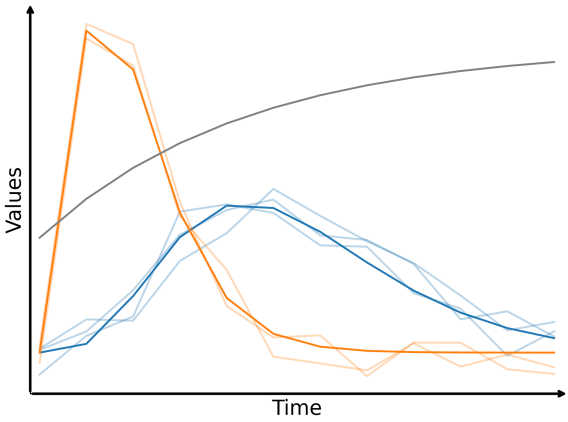

Fig. 1. An example of highlighting representative time series in a collection of 8 time series. A number of $\kappa = 2$ non-overlapping groups are selected and a time series in each group is selected as the time series representatives, as depicted in the emphasized blue and orange colors. The represented time series are also highlighted in the lighter colors. Hence, the selected time series cover the largest number of time series.

I. Introduction

Time series visualization is essential for many applications, such as meteorological analysis [29], [37], [47], urban planning [37], physiological signal processing [8], [27], [41], among others [15], [16], [44], [46]. Hence, many basic techniques for visualizing time series, such as timebox and density plot, have been widely investigated [1]–[3]. Although these tools and widgets have proven useful, recent advances in time series research demand more powerful and responsive time series visualization systems. In this paper, we address a common use scenario where analysts wish to visualize a collection of time series, freely select specific time series of interests for further inspection, and retrieve similar time series to those selected. An interactive visualization system can be highly beneficial in this scenario. For instance, consider an energy supply company that collects and generates numerous electricity usage time series from its millions of client households. Analysts want to find what patterns most electricity usage time series follow, for these representative usage time series aid the downstream tasks, including load scheduling optimization and pricing strategy development. An interactive visualization system can help answer this question by finding the most representative time series in a specific period of interest as well as highlight similar time series. This paper aims to address two primary challenges in building such an interactive visualization system. (a) Given a large collection of time series, how to efficiently select the most representative time series? More specifically, we want to select a small number, e.g., κ, of non-overlapping groups of time series. Time series within each group should be highly similar to each other in the same group. For each group, we also select a time series representative that represents the group,

aiming for these κ non-overlapping groups to represent or cover as many time series as possible. Fig. 1 depicts an example of choosing $\kappa = 2$ time series representatives as well as their represented groups from a collection of eight time series. The selection process should be efficient and fast, and ideally has no latency or delay. (b) When time series have missing data, how can we incorporate these incomplete time series during interactions? More specifically, how to render incomplete time series visualization and handle them during time series selection? An ideal visualization should not only clearly demonstrate the missing data but also take them into considerations when retrieving time series representatives.

To address the aforementioned challenges, we propose a novel time series visualization system, namely, **L**ocality Sensitive Hashing-based **I**nteractive **V**isualization **E**xplorer for Large-Scale **I**ncomplete **T**ime **S**eries (LIVE-ITS). LIVE-ITS is designed to visualize a large collection of incomplete time series efficiently. Moreover, our design allows analysts to select a time period of interest within which LIVE-ITS finds and highlights the most representative time series and indicates which each of these represent. The key contributions of our approach are outlined below,

- **Large-Scale Datasets.** LIVE-ITS efficiently visualizes large-scale time series datasets. The visualization is fast and responsive, with a time complexity that scales linearly with the number of time series;
- **Incomplete Time Series.** LIVE-ITS supports visualization for incomplete time series, where missing data are highlighted and taken into account upon selection queries;
- **Optimal Subset.** Upon queries, LIVE-ITS instantly retrieves a subset of the most representative time-series, as well as the series they represent. This subset represents as many time series as possible that belong to a user-selected κ number of non-overlapping groups. Notably, these representatives do not necessarily cover all time series, allowing for the presence of outlier time series.

Our paper is structured as follows: We begin with a brief review of closely related work in Sec. II. We then describe our visualization design in Sec. III, which covers both static presentation and user interactions. Next, Sec. IV details our proposed LSH-based algorithm. Specifically, Sec. IV-A explains the procedure for selecting time series representatives through the problem formulation of *time series subset selection*. A proof is provided to show that our proposed partition-based greedy algorithm yields an optimal subset, representing as many time series as possible. The subset selection requires a time series partitioning, which, as detailed in Sec. IV-B, is instantiated using the Locality-Sensitive Hashing (LSH) method. In Sec. V, we provide a time complexity analysis demonstrating that our method achieves the best possible time complexity. In Sec. VI, we design a series of experiments to validate the effectiveness and efficiency of our LIVE-ITS. In Sec. VII, an example is given to demonstrate how LIVE-ITS can be used in a realistic scenario. Lastly, we summarize our paper and point out potential future directions, as in Sec. VIII.

II. Related Work

In this section, we briefly review closely related work on large-scale time series visualization as well as subset selection.

Large-Scale Time Series Visualization. Many methods have been proposed for visualizing time series, particularly for large-scale time series datasets [8], [11], [15], [28], [31], [35], [40], [47]. The objectives of existing methods span a wide spectrum, including level-of-detail control [40], inter-relations exploration between time series [11], [28], backend system design [15], and data management [30], [31], among other aspects [1], [5], [6], [9], [36], [42]. In this paper, our primary objective is to develop an efficient interactive system for visualizing large-scale incomplete time series. Earlier research attempts include timesearcher [20], which utilizes timebox widgets; line graph explorer [22], which is based on focus and context; and other approaches utilizing bitmaps [23]. However, they are not optimized for selecting representative time series from large-scale datasets. More recent efforts focus on large-scale datasets. For example, Plotly-resampler [41] performs series-wise aggregation for scalability at the cost of inducing data point loss. StreamStory [37] leverages machine learning techniques to mine existing relations between in large-scale time series. Perhaps the closest work to our LIVE-ITS is KD-Box [47], which leverages a KD-tree structure with curve density estimates. In contrast to KD-Box, which bundles time series into a density field. Our approach visualizes each time series individually, allowing for more refined selection control. KD-Box requires pre-computing of curve decomposition and tree construction, whereas LIVE-ITS avoids these steps, making it more lightweight and efficient. In the context of incomplete time series visualization, related work includes Stroscope [8], which leverages a ripple graph to visualize irregularly measured time series. ViTST [25] takes a different approach by converting irregular time series to images. To the best of our knowledge, none of the existing work directly targets incomplete time series visualization; therefore, our proposed LIVE-ITS fills the emptiness in visualization systems for incomplete time series.

Subset Selection. Finding a representative subset from a larger pool of items is crucial to many applications, such as video summarization and scene categorization, among others [4], [12], [13], [32]. In the case of video summarization, the goal of subset selection is to choose a few screenshots or frames that capture the messages these videos convey. In contrast, our time series application aims to select a subset of time series that represents as many time series from the original set as possible. Moreover, we impose a constraint on the number of time series within the selected subset, that they should belong to one of several non-overlapping groups. Each of these groups contains time series that are highly similar to each other. A straightforward solution is to compute a time-series clustering and select a time series representative from each resulting cluster to construct the subset [19], [24], [26], [39]. However, this approach is computationally expensive and introduce high latency in visualization rendering. In comparison, LIVE-ITS

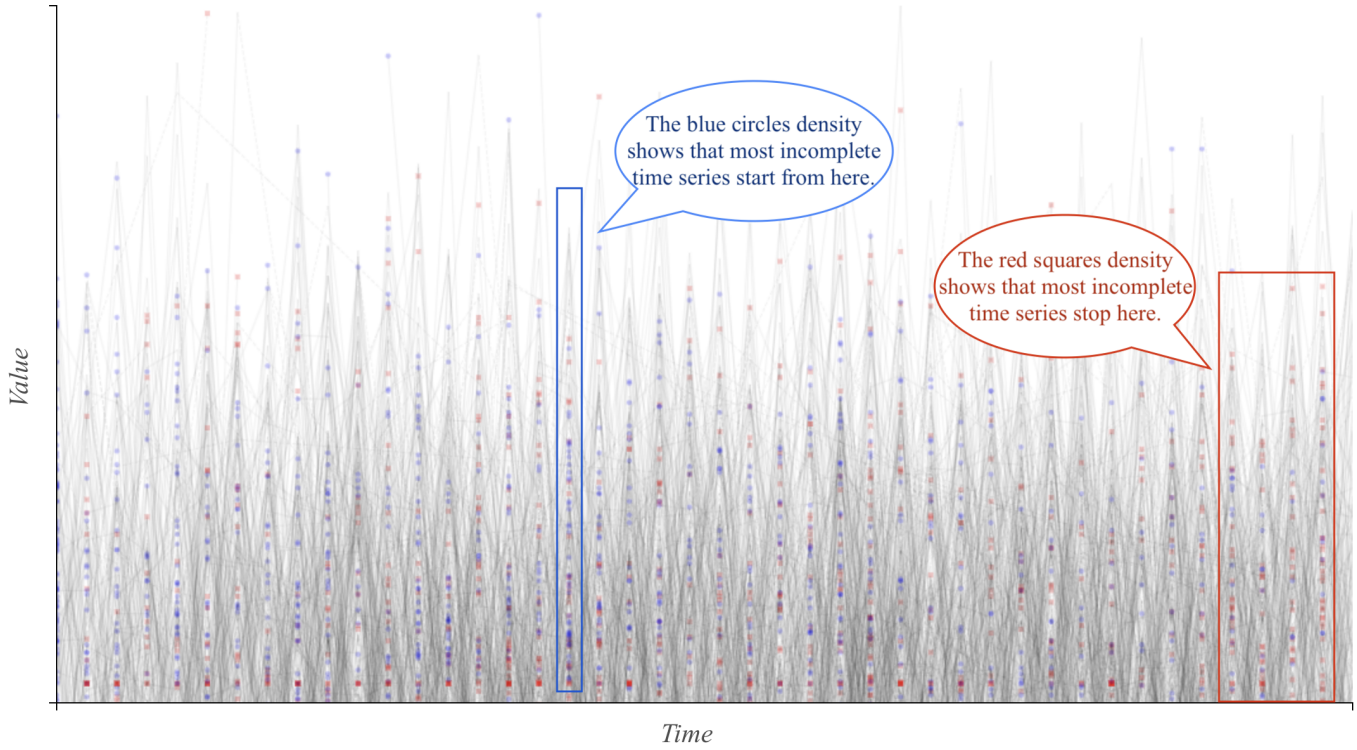

Fig. 2. An example of the static presentation of LIVE-ITS using synthetic data. The blue circles and red squares indicate the starting and ending points of the time series, respectively. The dashed lines highlight the missing values in the incomplete time series.

leverages time series partitioning through Locality-Sensitive Hashing, which has a low time complexity and enables quick rendering. The criteria for determining the best subset also vary across different papers [17], [38], [43]. For example, [47] utilizes time series density scores to select a subset that best represents the time series trends. Another line of work formulates the subset selection problem as a row-sparsity regularized trace minimization problem to find a subset that can represent all items in the target set [17]. This is a NP-hard problem [13], [48], and is not practical for our use scenarios due to the presence of outlier time series. Hence, our approach aims to select a subset that represents as many time series as possible, without necessarily including all time series, allowing for the exclusion of outlier series.

III. Visualization Design

In this section, we provide an overview of our visualization design with respects to its static presentation and supported user interactions.

A. Static Visualization

Our design primarily employs a line graph, where each available time series is individually plotted [21]. The line graph features two axes: the x-axis displays the timeline, and the y-axis shows the series values [20]. Label ticks are properly added to both axes to indicate the time and the value of each time point. Additionally, we use the following visualization cues to enhance clarity.

Non-selected Time Series. To distinguish between non-selected (default) and selected time series, a basic line color is used for non-selected series, such as black or white, depending on the chosen color scheme.

Missing Time Series Values. To distinguish missing series values from existing series values, we interpolate missing values using a linear function based on their two nearest existing values. Line segments containing interpolated values are plotted with a dashed line style.

Time Series Starting & Ending Points. To contrast between different incomplete series, we mark the starting and ending points of each series. Specifically, starting points are indicated with blue circles, and ending points with red squares.

Fig. 2 illustrates an example of our static visualization. The figure clearly shows a collection of time series, with missing values highlighted in interpolated line segments, plotted as dashed lines. Note that the starting and ending points marked by blue circles and red squares provide viewers density information about where most series start and end.

B. User Interactions

Our interaction design allows users to select an area and then respond with a set of time series representatives along with time series they represent. For this purpose, we employ two primary interactions: area selection and area modification.

Area Selection. Our area selection utilizes an interactive timebox [20], where users can click and drag to select an area in the line graph, as shown in Fig. 3 (left). The rectangular

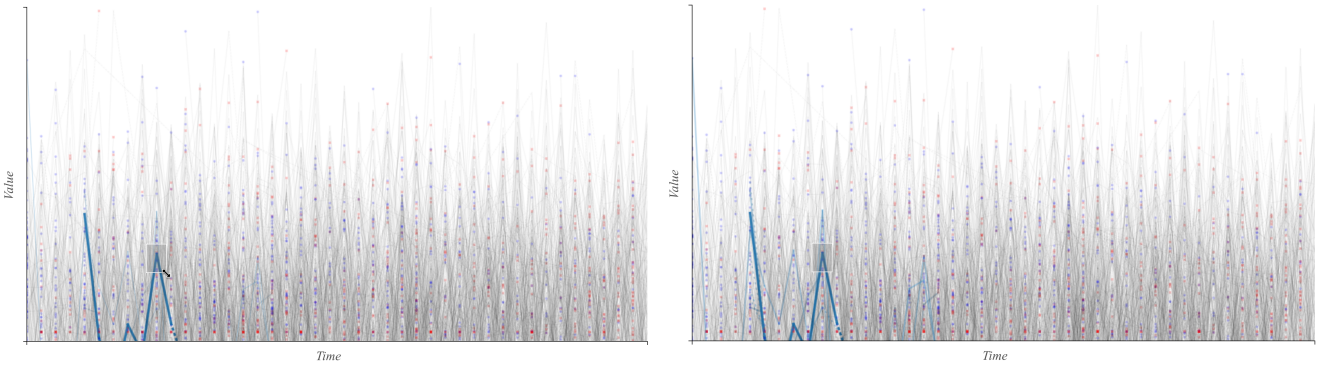

Fig. 3. An example of user interactions with area selection and area modification. Upon moving selected area or changing its boundaries, time series representatives are highlighted. When the movement is finished, represented time series are semi-highlighted.

selection is defined by four boundaries: *left*, *right*, *top*, and *bottom*. A time series is considered selected if its sub-series within the time interval of $[left, right]$ falls within the range $[top, bottom]$. In other words, a time series is selected if its sub-series within the time interval is entirely contained within the selected timebox. *Interaction Response*: when one or more time series are selected, an LSH-based algorithm is executed to choose a subset of these selected time series as representatives. The algorithm, as described in Sec. IV-A, facilitate the quick response of LIVE-ITS. These time series representatives are highlighted with various colors to distinguish them from non-selected time series. When the click event that triggers the selection is released, time series similar to the time series representatives are also highlighted with semi-transparent colors, as shown in Fig. 3 (right).

Area Modification. If an area selection (timebox) already exists, analysts can modify the area by moving the timebox or adjusting its boundaries. *Interaction Response*: When the timebox is moved or its boundaries are changed, the set of selected time series gets updated, leading to changes in the time series representatives. Time series similar to the representatives will be highlighted upon click release event to guarantee an efficient and responsive rendering.

IV. Backend Algorithms

This section begins with a general problem formulation of subset selection, which aims to choose a subset from a source set to represent a target set. We then propose a greedy algorithm and prove that it guarantees optimal subset selection. In other words, our proposed greedy algorithm generates a subset of time series representatives that covers as many time series as possible. Our greedy algorithm relies on time series partitioning; therefore, we propose using Locality-Sensitive Hashing (LSH) for this purpose [10], [33]. We provide a time complexity analysis to demonstrate the responsiveness advantage of our LSH-based approach.

A. Time Series Subset Selection

Time series subset selection aims to select a subset S of at most κ time series from a source set $X = \{X_1, X_2, \ldots, X_N\}$ to represent a target set $Y = \{Y_1, Y_2, \ldots, Y_M\}$, where $N = |X|$ and $M = |Y|$ denote the total number of time series in X and Y, respectively. In our case, we also require that the source set be a subset of the target set, as $X \subseteq Y$. Assume that we have a non-overlapping partition of Y into K groups, where each group contains highly similar time series. Let $D \in \{0,1\}^{K \times M}$ be a binary group membership matrix of Y, hence,

$$D_{km} = \begin{cases} 1, & \text{if } k\text{(th) group contains } Y_m \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

By definition, each time series Y_m appears in exactly one of the K groups, which implies that each column of D sums to 1, $\sum_k D_{km} = 1,\ m \in \{1, 2, \ldots, M\}$. Note that Y may have outlier series, so it is not guaranteed that a subset of X will represent all series in Y. In this case, we opt to select a subset that represents as many series in Y as possible. Therefore, we reorder the K partitioned groups of D from largest to smallest group size, denoted by $G = \{G_1, G_2, \ldots, G_K\}$ with $|G_1| \geq |G_2| \geq \ldots \geq |G_K|$ and $|G_1| \cup |G_2| \cup \ldots \cup |G_K| = M$. Let S be initialized as an empty set $S = \{\}$. We iterate through $G_1, G_2, \ldots$, and for each group G_k, the set S is updated based on the number of source series G_k contains,

$$S = \begin{cases} S, & \text{if } |G_k \cap X| = 0. \\ S \cup \mathrm{Repr}\,(G_k)\,, & \text{if } |G_k \cap X| \geq 1. \end{cases} \tag{2}$$

where $\mathrm{Repr}\,(\cdot)$ chooses a representative time series from the group G_k. There are many eligible criteria for implementing $\mathrm{Repr}\,(\cdot)$. For example, using the centroid or median time series can produce a good visualization. In our algorithm, we opt for the time series X_i that has the minimum Euclidean distance to all other time series, as defined below,

$$\mathrm{Repr}\,(G_k) = \underset{X_i}{\mathrm{argmin}} \sum_j \sum_{t=1}^{T} \sqrt{\left(X_i^t - X_j^t\right)^2},\ X_j \in G_k \tag{3}$$

The iteration stops once S reaches a user-defined number, κ, of time series. A concise summary of the computational procedure is outlined in Algorithm 1.

When a series from G_k is included in S, it represents all other series in G_k. As a result, S contains at most κ series

Algorithm 1 Selecting a Representative Time-Series Subset

Input: K groups of time series $G = \{G_1, G_2, \ldots, G_K\}$ with $|G_1| \geq |G_2| \geq \ldots \geq |G_K|$, a set of time series representative candidates X, and a representative number limit κ;
Initialize an empty set $S = \{\}$
for each group G_i, $i = 1, 2, \ldots$ **do**
  if $|G_k \cap X| \geq 1$ **then**
    $\text{Repr}(G_k) \leftarrow \text{argmin}_{X_i} \sum_j \sum_{t=1}^{T} \sqrt{\left(X_i^t - X_j^t\right)^2}$
    $S \leftarrow S \cup \text{Repr}(G_k)$
  end if
  if $|S| = \kappa$ **then**
    Exit the loop {Early stop}
  end if
end for
Output: selected time series representatives S

from X that represent as many time series as possible from Y. We prove by contradiction that S is the **optimal subset**.

Proof. Assume there exist two series $X_i, X_j \in X$, where $X_i \in S$ is selected and $X_j \notin S$ is not selected. By our definition, $X_i \in G_i$ represents all time series in G_i, and $X_j \in G_j$ represents all time series in G_j. We aim to prove that X_i cannot be replaced by X_j to increase the number of series represented by S.

To prove by contradiction, we first assume that X_j can replace X_i to increase the number of represented series in S, i.e., $|G_i| < |G_j|$, and then show that this assumption leads to a contradiction. In Eq. 2, larger groups are ranked before smaller groups. Hence, if the assumption stands, the set G_j will be considered before G_i. On the one hand, if the subset size limit κ is already reached before considering G_j, the series X_i will not be added, as the iteration would stop before G_i is considered. This contradicts the premise that $X_i \in G_i$ is selected. On the other hand, if the subset size limit has not been reached, since G_j contains at least one series $X_j \in X$ from X, a representative series from X_j must be added to S to represent G_j. Adding the series X_j to S would contradict the premise that $X_j \notin S$.

In conclusion, there is no pair of series $X_i \in S$ and $X_j \notin S$ such that X_j can replace X_i to increase the number of represented time series. Hence, S is an optimal subset of X that represents the maximum number of time series from Y. □

In LIVE-ITS, we define Y as all time series in the dataset, and X as the selected time series within the timebox, or equivalently, the selected area.

B. LSH-based Partitioning

This section introduces a fast and efficient partitioning method for time series data, based on Locality-Sensitive Hashing (LSH) [7], [45]. Let $\mathbf{Y} \in \mathbb{R}^{M \times T}$ denote a collection of M time series, each of length T. We leverage LSH to generate a signature for each time series and then allocate time series with identical signatures into the same bins. During the signature generation phase, a random matrix $\mathbf{Y}^{rand} \in [-1, 1]^{T \times b}$ is created to transform the time series into b-dimensional vectors $\mathbf{Y}^b = \mathbf{Y}\mathbf{Y}^{rand} \in \mathbb{R}^{M \times b}$. Here, b is a user-selected parameter that controls the length of the signatures. The b-dimensional vectors are then converted to signatures through a sign function:

$$\mathbf{Y}_{ij}^{sig} = \begin{cases} 1, & \text{if } \mathbf{Y}_{ij}^b \geq 0 \\ 0, & \text{otherwise} \end{cases} \in \mathbb{R}^{M \times b} \tag{4}$$

Hence, there are at maximum 2^b different signatures from $\underbrace{00\ldots0}_{b-\text{digits}}$ to $\underbrace{11\ldots1}_{b-\text{digits}}$, each corresponding to a unique partition.

Missing Values. To mitigate the effects of missing values in time series partitioning, we set time points with missing values in $\mathbf{Y}$ to zeros, so they do not contribute to $\mathbf{Y}^b$. Alternatively, using interpolated values is another option for grouping similar time series.

V. Time Complexity Analysis

In this section, we analyze the time complexity of our efficient LSH-based time series visualization. The time complexity can be decomposed into three stages: partitioning, querying, and visualization.

LSH partitioning stage. It takes $\mathcal{O}(Tb)$ to generate the random matrix $\mathbf{Y}^{rand} \in \mathbb{R}^{T \times b}$, $\mathcal{O}(MTb)$ to product the time series matrix $\mathbf{Y} \in \mathbb{R}^{M \times T}$ with the random matrix $\mathbf{Y}^{rand} \in \mathbb{R}^{T \times b}$, and $\mathcal{O}(Mb)$ to generate the signature matrix $\mathbf{Y}^{sig} \in \mathbb{R}^{M \times b}$. The overall time complexity for partitioning is

$$\text{TC}_{partition} = \mathcal{O}(Tb) + \mathcal{O}(MTb) + \mathcal{O}(Mb) = \mathcal{O}(MT)$$

with b being a small selected parameter denoting the length of signatures, which can be considered a negligible constant.

Querying stage. We aim to select at most κ series from M time series in X to represent as many series in Y (equivalently, $\mathbf{Y}$) as possible. Let K denote the number of unique partitioned groups from the LSH partitioning stage. Recall that $X \subseteq Y$, hence, it takes $\mathcal{O}(1)$ to find which group a series $X_i \in X$ belongs to, as well as the size of the corresponding group. Moreover, it takes $\mathcal{O}(MlogM)$ to sort M time series in X based on their corresponding group size. As stated in Sec. IV-A, we iterate the re-ordered M time series and their groups, and construct the subset S using Eq. 2. If a group has one time series from X, it takes $\mathcal{O}(1)$ to compute $S \cup \text{Repr}(G_k)$. If a group has two or more time series from X, it takes $\mathcal{O}(T|G_k| + T)$ to compute $S \cup \text{Repr}(G_k)$. Since $\sum_G |G_k| = M$, it takes at most $\text{TC}_{query} = \mathcal{O}(MT)$ for each query. This time complexity describes the worst-case scenario; in practice, it is much faster due to the early stopping of the query when $|S|$ reaches the limit κ.

Visualization stage. For the static presentation, it takes $\text{TC}_{visualization} = \mathcal{O}(NT)$ to plot all N time series, each of length T.

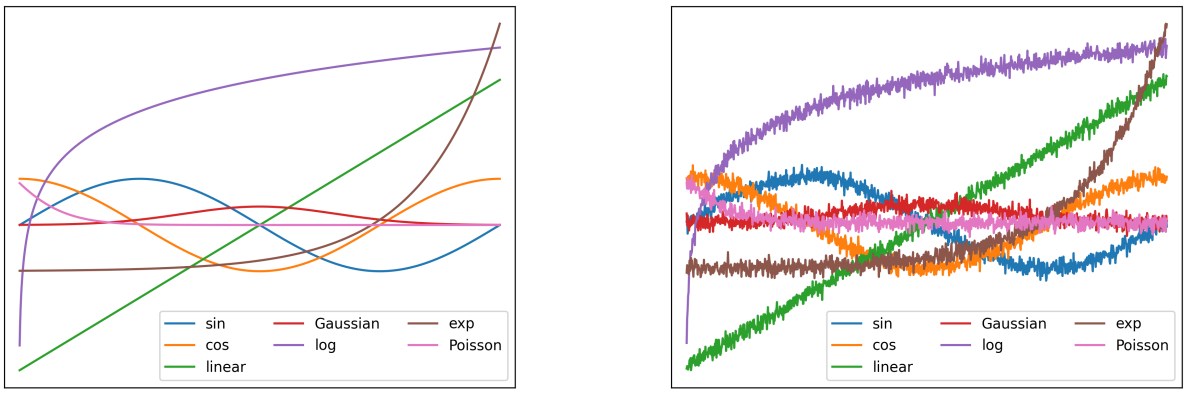

Fig. 4. An example of seven time series from the seven distributions (left), and the resulting time series with the white noise (right), in the synthetic dataset.

TABLE I
A SUMMARY OF DATA STATISTICS.

Dataset	#Series	Min	Median	Max	Missing Rate
SYNTHETIC	$70k$	-3.56	0.10	4.77	0%
WIKIPEDIA	$123k$	0	88	9999	2.34%

The overall time complexity is

$$\begin{aligned} \mathrm{TC}_{total} &= \mathrm{TC}_{partition} + \mathrm{TC}_{query} + \mathrm{TC}_{visualization} \\ &= \mathcal{O}(MT) + \mathcal{O}(MT) + \mathcal{O}(NT) \\ &= \mathcal{O}(NT) \end{aligned} \tag{5}$$

with $M < N$. This is in theory the best time complexity any algorithm can achieve as even storing and plotting all time series in Y take at least $\mathcal{O}(NT)$.

VI. EXPERIMENTS

To validate the effectiveness and efficiency of our proposed LIVE-ITS, we design a series of experiments using both a synthetic dataset and a real-world dataset. With these datasets, we aim to answer the following research questions:

a Does the LSH-based algorithm accurately partition similar time series into the same bins? Or equivalently, are time series in the partitioned bins similar to one another within the same bin?
b What is the time-series coverage of the LSH-based algorithm over the used datasets?
c What is the runtime performance of the proposed LIVE-ITS system? Specifically, is it scalable to handle large-scale time-series datasets?

Datasets. We utilize two datasets: the first dataset is a SYNTHETIC dataset generated from seven different distributions, including sine, cosine, linear, exponential, log, Gaussian, and Poisson. We carefully select involved parameters of each distribution function, such as offset, scale, and variance, to ensure that the resulting time series are within a similar range.

Moreover, we follow existing practices by adding Gaussian white noise to these base functions [18], [21], [47]. Fig. 4 shows an example of seven generated time series, each derived from one of the seven distributions below,

$$y(t) = T(t) + \epsilon \tag{6}$$

where T is generated from one of the seven base functions and ϵ denotes the white noise component sampled from a Gaussian distribution $\mathcal{N}(\mu = 0, \sigma = 0.1)$:

$$T(t) = \begin{cases} \text{Sin} & : \sin(t) \\ \text{Cos} & : \cos(t) \\ \text{Linear} & : t - \pi \\ \text{Gaussian} & : \mathcal{N}(\mu = \pi, \sigma = 1) \\ \text{Log} & : \ln(t + 0.01) + 2 \\ \text{Exponential} & : 0.01 * e^{(t)} - 1 \\ \text{Poisson} & : \frac{e^{-0.1} * 0.1^t}{t!} \end{cases}$$

We generate $10,000$ time series from each base function in the domain of $t \in [0, 2\pi]$, resulting in a total of $70,000$ time series. Time series are sampled to include $1,000$ time points.

The second dataset is a real-world dataset called WIKIPEDIA, which consists of web traffic data [34]. The dataset contains more than $140,000$ time series of Wikipedia website views over a span of one and a half years. We concentrate on the data from the first month and remove time series with only missing values or those with large values, resulting in a total of $123,085$ time series. A summary of the data statistics is presented in Table I.

A. Partitioning Accuracy

To answer research question (a), i.e., to validate the accuracy of the LSH-based algorithm, we apply the algorithm to the SYNTHETIC dataset. The algorithms generates bins of similar time series. As a reminder, the dataset contains $10,000$ time series from each of the seven distributions.

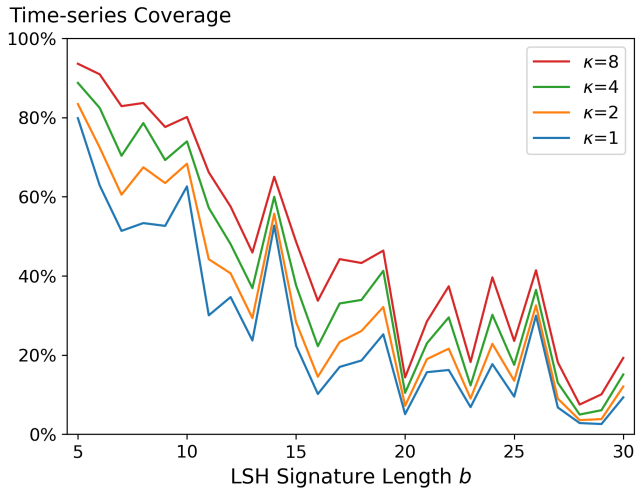

Fig. 5. Time-series coverage of the top $\kappa \in \{1, 2, 4, 8\}$ bins on various LSH signature length $b \in [5, 30]$. The time-series coverage has a decreasing trend as the LSH signature length b increases.

Ideally, time series from the same distribution function should be assigned to the same bins. In other words, the purer each bin is, the more accurate the partitioning results will be. To evaluate the partitioning accuracy, we compute the overall entropy loss of all bins as a linear combination of the entropy loss from each bin, according to their sizes:

$$Loss = \sum_{i=1}^{2^b} \frac{1}{|B_i|} \text{entropy}\,(B_i) \tag{7}$$

where $B = \{B_1, B_2, \ldots, B_{2^b}\}$ denotes the set of 2^b bins. $\text{entropy}\,(\cdot)$ denotes the entropy loss function, and is computed for each bin B_i:

$$\text{entropy}\,(B_i) = -\sum_j q_{ij} \log_2 q_{ij} \tag{8}$$

where q_{ij} denotes the label percentage of j(th) distribution.

We compute the *Loss* value of the LSH-based algorithm on the SYNTHETIC dataset, which yields a value of 0.35%. This value is very close to zero, indicating the high purity of most bins and demonstrating the accuracy of the LSH-based partition algorithm.

B. Time-series Coverage

To answer research question (b), we investigate the time-series coverage of LIVE-ITS, defined as the proportion of selected time series relative to the total number of time series. We apply the partition algorithm to the WIKIPEDIA dataset, which contains $123,085$ time series. We vary the LSH signature length parameter b from 5 to 30. For each value of b, we compute the time-series coverage for the largest κ groups, where $\kappa \in \{1, 2, 4, 8\}$. Fig. 5 illustrates the resulting time series coverage.

Notably, as the LSH signature length b increases, the time-series coverage has a decreasing trend. This is because a larger b leads to a more refined grouping and separation of time series. Another observation is that the time-series coverage of the largest group remains dominantly large, as indicated by the blue line in the figure. Meanwhile, the second to the eighth largest groups do not contribute as much coverage. This reflects the fact that most website traffic time series in the Wikipedia dataset are highly similar to one another.

TABLE II
RESPONSE TIME OF LIVE-ITS ON DATASETS OF VARIOUS SIZES.

#Time series	Mean ± Std. Time (ms)	
	Backend Loading	Frontend Rendering
$1,000$	15 ± 5.4	13.7 ± 3.0
$2,000$	26 ± 6.5	22.1 ± 4.3
$4,000$	51 ± 9.5	45.3 ± 7.7
$8,000$	95 ± 8.5	90.9 ± 30.9
$16,000$	271 ± 9.9	223.8 ± 34.7
$32,000$	872 ± 54.0	502.7 ± 56.0
$64,000$	$3.1s \pm 73.1$	1065.8 ± 128.9
$123,085$	$10.5s \pm 47.9$	2027.8 ± 247.8

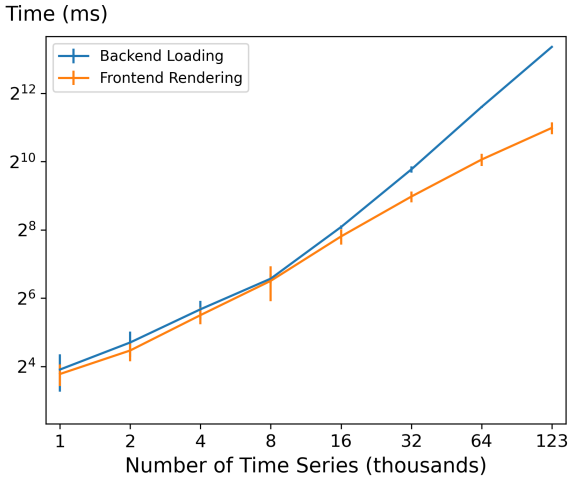

Fig. 6. The response time of LIVE-ITS on datasets of various sizes. Both backend loading and frontend rendering time are efficiently linear to the number of time series.

C. Efficiency Performance

To answer research question (c), we evaluate the efficiency of LIVE-ITS by measuring the response time across datasets of various sizes. Specifically, we measure the backend loading time and the frontend rendering time on datasets ranging from $1,000$ time series to $123,085$ time series, with an exponentially increasing step of $2\times$.

The backend loading time is defined as the time Python takes to load the time series data files. The frontend rendering time is measured as the duration between the start of an interaction (such as area selection or modification) and the completion of time-series plotting.

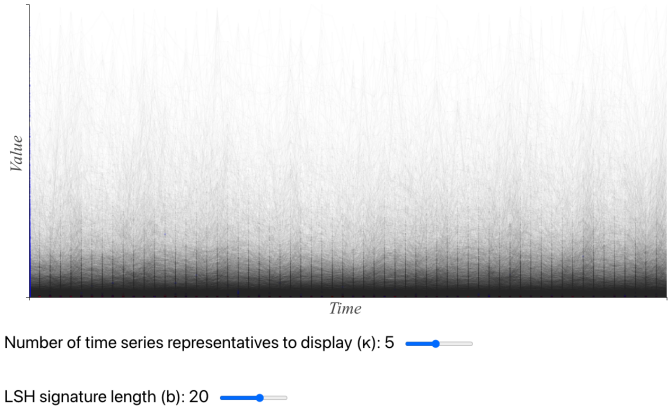

Fig. 7. Step 1 of 5: Initial rendering of LIVE-ITS, where a static presentation of time-series is exhibited.

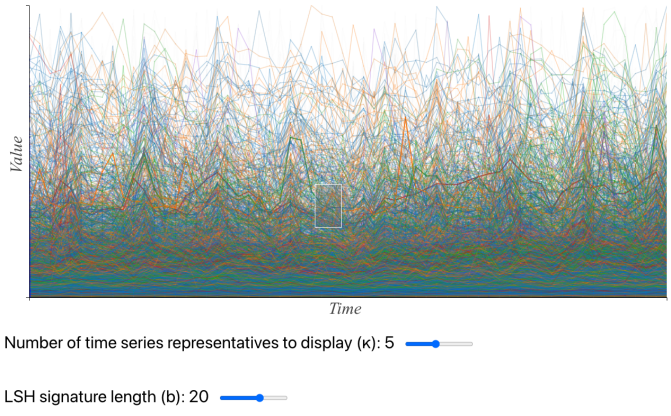

Fig. 8. Step 2 of 5: Initial response upon query, where a default number ($\kappa = 5$) of time series groups are highlighted.

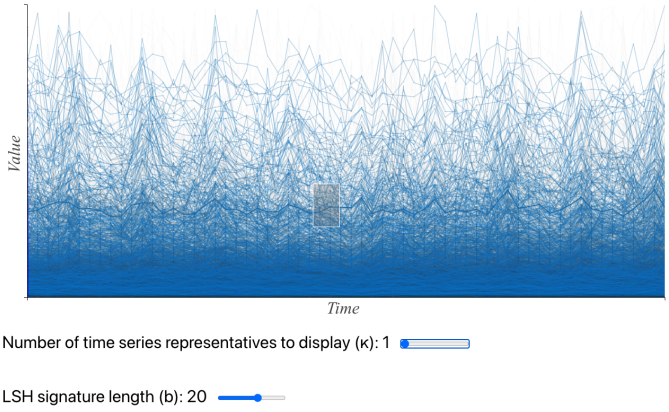

Fig. 9. Step 3 of 5: Response after setting $\kappa = 1$, where the most representative time series group is highlighted.

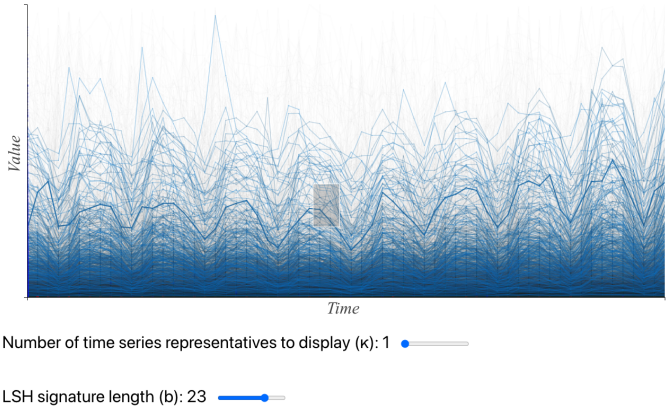

Fig. 10. Step 4 of 5: Response after setting $b = 23$, where the most representative time series group is re-calculated base on the LSH algorithm, giving ideal results.

We repeat 10 trials for each experiment and report the mean and the standard deviation of response time, as shown in Table II and Fig. 6. We observe that both the backend loading time and the frontend rendering time are steadily proportional to the dataset size, exhibiting the linear time complexity.

We also notice that the frontend rendering time is faster than the backend loading time, and scales better to the growth of the datasets, which proves the scalability of the visualization. ***Machine Specifications.*** All experiments are conducted on the backend of a *gunicorn* web server with 10 workers, as well as within the *npm* production environment. By default, the LSH signature length is set to 10, and the number of shown time-series representatives is set to 5.

The machine specifications include a 12-cores CPU and 18GB of memory on a Mac15.6 model equipped with the Apple M3 Pro chip. Our interaction system is implemented using *d3.js*, *node.js*, *React*, and *Flask* [14]. We also utilize a Docker environment to support the package management.

VII. Case Study

In this section, we conduct a case study to demonstrate how LIVE-ITS can be used in a realistic scenario. We render $16,000$ time series over two months of Wikipedia traffic data. Fig. 7 shows the static presentation. As mentioned earlier, we allow users to control two parameters: the number of time series representatives, κ, and the LSH signature length, b.

To identify representative time series, an analyst may select an area of interests by creating a timebox on the line plot, as shown in Fig. 8. However, the visualization is difficult to interpret, with too many highlighted time series overlapping one another. This occurs because the parameter κ is set too high, with a default value of 5.

The analyst can then interact with LIVE-ITS and reduce κ to 1 to investigate the most dominant time series patterns, which results in Fig. 9. However, the resulting visualization is still not ideal due to the lack of clear dominant time series patterns. In other words, dissimilar time series appear to be grouped into the largest bin. To address this, the analyst can increase the LSH signature length b to restrict each bin to grouping only more similar time series.

For example, after setting $b = 23$, the resulting visualization exhibits intuitive time series seasonality, as shown in Fig. 10. The analyst can further investigate related information about

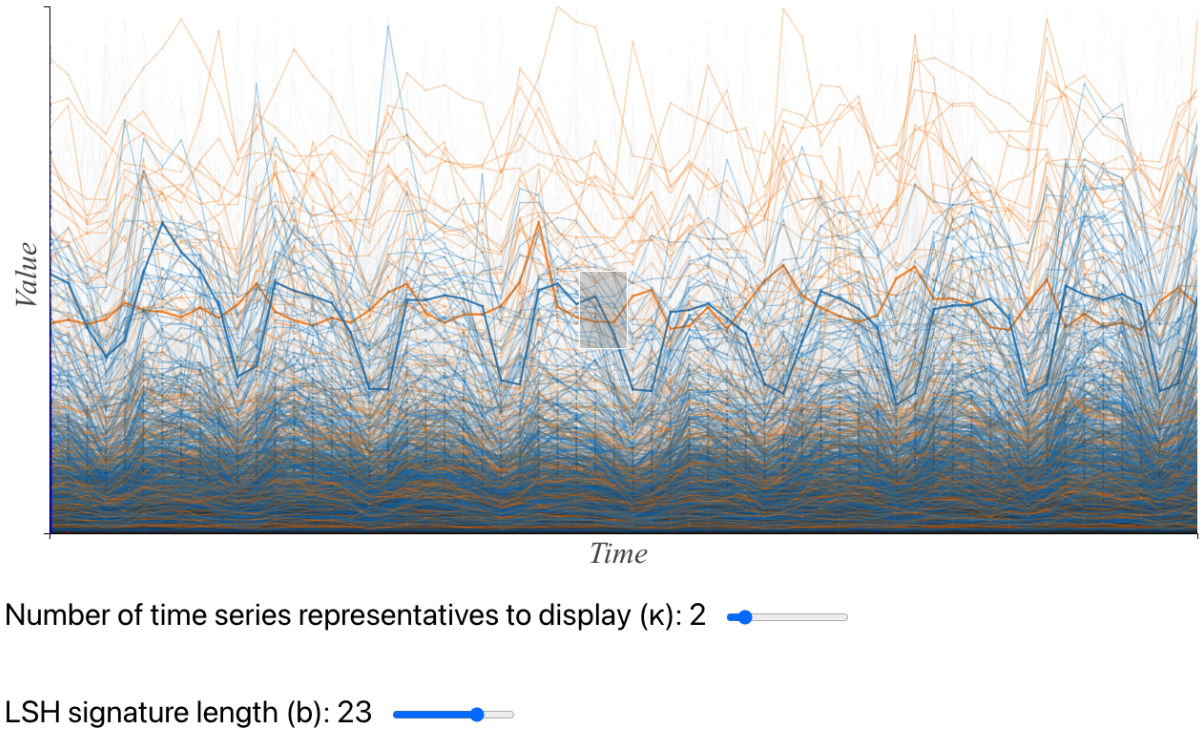

Fig. 11. Step 5 of 5: Response after setting $\kappa = 2$, where the secondary most representative group is also highlighted.

the selected time series. In our dataset, each time series is associated with a web page. We find that most highlighted time series correspond to connected celebrities.

Furthermore, the analyst can validate whether the second largest group exhibits a different time series pattern by increasing κ and adjusting the timebox, as shown in Fig. 11.

VIII. Conclusion

In this paper, we address the task of time series visualization. We propose a novel time series interaction explorer called LIVE-ITS to tackle existing challenges of visualizing large-scale incomplete time-series data. Theoretical proof is provided to demonstrate that LIVE-ITS yields an optimal subset containing the most representative time series, with the best possible time complexity.

To evaluate the effectiveness of LIVE-ITS, we design a series of experiments to test its partitioning accuracy and time-series coverage. Moreover, we assess the scalability of LIVE-ITS by applying it to datasets of various sizes. The results demonstrate that our system exhibits runtime growth proportional to the number of time series. A case study is also conducted to showcase the use of LIVE-ITS.

In the future, We plan to extend this work to enable more comprehensive time series analyses and empirical applications, including time series labelling and clustering. Additionally, LIVE-ITS can be customized with various features and functions to accommodate the diverse needs of analysts, such as the manual addition and removal of specific time series, and the presentation of relevant information through a tooltip or other widgets. We also aim to explore alternative methods for managing missing time points. With the capacities of LIVE-ITS, we anticipate it will prove valuable to analysts working with incomplete large-scale time series data.

References

[1] Muhammad Adnan, Mike Just, and Lynne Baillie. Investigating time series visualisations to improve the user experience. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 5444–5455, 2016.

[2] Wolfgang Aigner, Silvia Miksch, Wolfgang Müller, Heidrun Schumann, and Christian Tominski. Visual methods for analyzing time-oriented data. *IEEE transactions on visualization and computer graphics*, 14(1):47–60, 2007.

[3] Wolfgang Aigner, Silvia Miksch, Heidrun Schumann, and Christian Tominski. *Visualization of time-oriented data*, volume 4. Springer, 2011.

[4] Aws Naser Jaber Al. Efficient visualization framework for real-time monitoring network traffic of high-speed networks. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 5839–5842. IEEE, 2021.

[5] Danielle Albers, Michael Correll, and Michael Gleicher. Task-driven evaluation of aggregation in time series visualization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 551–560, 2014.

[6] Daniel Braun, Rita Borgo, Max Sondag, and Tatiana von Landesberger. Reclaiming the horizon: Novel visualization designs for time-series data with large value ranges. *IEEE Transactions on Visualization and Computer Graphics*, 2023.

[7] Hongjie Chen, Ryan Rossi, Sungchul Kim, Kanak Mahadik, and Hoda Eldardiry. Evolving super graph neural networks for large-scale time-series forecasting. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 201–212. Springer, 2024.

[8] Myoungsu Cho, Bohyoung Kim, Hee-Joon Bae, and Jinwook Seo. Stroscope: Multi-scale visualization of irregularly measured time-series data. *IEEE transactions on visualization and computer graphics*, 20(5):808–821, 2014.

[9] Michael Correll, Danielle Albers, Steven Franconeri, and Michael Gleicher. Comparing averages in time series data. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1095–1104, 2012.

[10] Anirban Dasgupta, Ravi Kumar, and Tamás Sarlós. Fast locality-sensitive hashing. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1073–1081, 2011.

[11] Zikun Deng, Shifu Chen, Tobias Schreck, Dazhen Deng, Tan Tang, Mingliang Xu, Di Weng, and Yingcai Wu. Visualizing large-scale spatial time series with geochron. *IEEE Transactions on Visualization and Computer Graphics*, 2023.

[12] Ehsan Elhamifar and M De Paolis Kaluza. Subset selection and summarization in sequential data. *Advances in Neural Information Processing Systems*, 30, 2017.

[13] Ehsan Elhamifar, Guillermo Sapiro, and S Shankar Sastry. Dissimilarity-based sparse subset selection. *IEEE transactions on pattern analysis and machine intelligence*, 38(11):2182–2197, 2015.

[14] Loraine Franke and Daniel Haehn. Modern scientific visualizations on the web. In *Informatics*, volume 7, page 37. MDPI, 2020.

[15] Manele Ait Habouche, Mickaël Kerboeuf, Goulven Guillou, and Jean-Philippe Babau. Fast: An efficient framework for visualizing large-scale time series. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 3745–3754. IEEE, 2022.

[16] Dongyun Han and Isaac Cho. Interactive visualization for smart power grid efficiency and outage exploration. In *2023 IEEE International Conference on Big Data (BigData)*, pages 656–661. IEEE, 2023.

[17] Hussein Hazimeh and Rahul Mazumder. Fast best subset selection: Coordinate descent and local combinatorial optimization algorithms. *Operations Research*, 68(5):1517–1537, 2020.

[18] Jeffrey Heer, Nicholas Kong, and Maneesh Agrawala. Sizing the horizon: the effects of chart size and layering on the graphical perception of time series visualizations. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1303–1312, 2009.

[19] Andrew Hill, Russell Bowler, Katerina Kechris, and Farnoush Banaei-Kashani. Semi-supervised embedding for scalable and accurate time series clustering. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 942–951. IEEE, 2022.

[20] Harry Hochheiser and Ben Shneiderman. Dynamic query tools for time series data sets: timebox widgets for interactive exploration. *Information Visualization*, 3(1):1–18, 2004.

[21] Waqas Javed, Bryan McDonnel, and Niklas Elmqvist. Graphical perception of multiple time series. *IEEE transactions on visualization and computer graphics*, 16(6):927–934, 2010.

[22] Robert Kincaid and Heidi Lam. Line graph explorer: scalable display of line graphs using focus+ context. In *Proceedings of the working conference on Advanced visual interfaces*, pages 404–411, 2006.

[23] Nitin Kumar, Venkata Nishanth Lolla, Eamonn Keogh, Stefano Lonardi, Chotirat Ann Ratanamahatana, and Li Wei. Time-series bitmaps: a practical visualization tool for working with large time series databases. In *Proceedings of the 2005 SIAM international conference on data mining*, pages 531–535. SIAM, 2005.

[24] Xiaosheng Li, Jessica Lin, and Liang Zhao. Time series clustering in linear time complexity. *Data Mining and Knowledge Discovery*, 35(6):2369–2388, 2021.

[25] Zekun Li, Shiyang Li, and Xifeng Yan. Time series as images: Vision transformer for irregularly sampled time series. *Advances in Neural Information Processing Systems*, 36, 2024.

[26] T Warren Liao. Clustering of time series data—a survey. *Pattern recognition*, 38(11):1857–1874, 2005.

[27] Jessica Lin, Eamonn Keogh, Stefano Lonardi, Jeffrey P Lankford, and Donna M Nystrom. Visually mining and monitoring massive time series. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 460–469, 2004.

[28] Lauro Lins, James T Klosowski, and Carlos Scheidegger. Nanocubes for real-time exploration of spatiotemporal datasets. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2456–2465, 2013.

[29] Shawn Martin and Tu-Toan Quach. Interactive visualization of multivariate time series data. In *Foundations of Augmented Cognition: Neuroergonomics and Operational Neuroscience: 10th International Conference, AC 2016, Held as Part of HCI International 2016, Toronto, ON, Canada, July 17-22, 2016, Proceedings, Part II 10*, pages 322–332. Springer, 2016.

[30] Peter McLachlan, Tamara Munzner, Eleftherios Koutsofios, and Stephen North. Liverac: interactive visual exploration of system management time-series data. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1483–1492, 2008.

[31] Fabio Miranda, Marcos Lage, Harish Doraiswamy, Charlie Mydlarz, Justin Salamon, Yitzchak Lockerman, Juliana Freire, and Claudio T Silva. Time lattice: A data structure for the interactive visual analysis of large time series. In *Computer Graphics Forum*, volume 37, pages 23–35. Wiley Online Library, 2018.

[32] Margaret Mitchell, Dylan Baker, Nyalleng Moorosi, Emily Denton, Ben Hutchinson, Alex Hanna, Timnit Gebru, and Jamie Morgenstern. Diversity and inclusion metrics in subset selection. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 117–123, 2020.

[33] Loïc Paulevé, Hervé Jégou, and Laurent Amsaleg. Locality sensitive hashing: A comparison of hash function types and querying mechanisms. *Pattern recognition letters*, 31(11):1348–1358, 2010.

[34] Navyasree Petluri and Eyhab Al-Masri. Web traffic prediction of wikipedia pages. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 5427–5429. IEEE, 2018.

[35] Stijn J Rotman, Boris Cule, and Len Feremans. Efficiently mining frequent representative motifs in large collections of time series. In *2023 IEEE International Conference on Big Data (BigData)*, pages 66–75. IEEE, 2023.

[36] Conglei Shi, Weiwei Cui, Shixia Liu, Panpan Xu, Wei Chen, and Huamin Qu. Rankexplorer: Visualization of ranking changes in large time series data. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2669–2678, 2012.

[37] Luka Stopar, Primoz Skraba, Marko Grobelnik, and Dunja Mladenic. Streamstory: exploring multivariate time series on multiple scales. *IEEE transactions on visualization and computer graphics*, 25(4):1788–1802, 2018.

[38] Ryan Thompson. Robust subset selection. *Computational Statistics & Data Analysis*, 169:107415, 2022.

[39] Sandhya Tripathi, N Hemachandra, and Prashant Trivedi. Interpretable feature subset selection: A shapley value based approach. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 5463–5472. IEEE, 2020.

[40] Yumiko Uchida and Takayuki Itoh. A visualization and level-of-detail control technique for large scale time series data. In *2009 13th International Conference Information Visualisation*, pages 80–85. IEEE, 2009.

[41] Jonas Van Der Donckt, Jeroen Van Der Donckt, Emiel Deprost, and Sofie Van Hoecke. Plotly-resampler: Effective visual analytics for large time series. In *2022 IEEE Visualization and Visual Analytics (VIS)*, pages 21–25. IEEE, 2022.

[42] James Walker, Rita Borgo, and Mark W Jones. Timenotes: a study on effective chart visualization and interaction techniques for time-series data. *IEEE transactions on visualization and computer graphics*, 22(1):549–558, 2015.

[43] Kiyoung Yang, Hyunjin Yoon, and Cyrus Shahabi. A supervised feature subset selection technique for multivariate time series. In *Proceedings of the workshop on feature selection for data mining: Interfacing machine learning with statistics*, pages 92–101. Citeseer, 2005.

[44] Matthew Young, Sangmi Pallickara, and Shrideep Pallickara. Aqua: A framework for spatiotemporal analysis and visualizations of water quality data at scale. In *2023 IEEE International Conference on Big Data (BigData)*, pages 1555–1562. IEEE, 2023.

[45] Yuncong Yu, Dylan Kruyff, Jiao Jiao, Tim Becker, and Michael Behrisch. Pseudo: Interactive pattern search in multivariate time series with locality-sensitive hashing and relevance feedback. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):33–42, 2022.

[46] Xuchao Zhang, Yifeng Gao, Jessica Lin, and Chang-Tien Lu. Tapnet: Multivariate time series classification with attentional prototypical network. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 6845–6852, 2020.

[47] Yue Zhao, Yunhai Wang, Jian Zhang, Chi-Wing Fu, Mingliang Xu, and Dominik Moritz. Kd-box: Line-segment-based kd-tree for interactive exploration of large-scale time-series data. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):890–900, 2021.

[48] Xiaobin Zhi and Yigang Qi. An improved dissimilarity-based sparse subset selection algorithm. In *2022 4th International Conference on Natural Language Processing (ICNLP)*, pages 611–615. IEEE, 2022.