TSUBASA: Climate Network Construction on Historical and Real-Time Data

ABSTRACT

A climate network represents the global climate system by the interactions of a set of anomaly time-series. Network science has been applied on climate data to study the dynamics of a climate network. The core task and first step to enable interactive network science on climate data is the efficient construction and update of a climate network on user-defined time-windows. We present TSUBASA, an algorithm for the efficient construction of climate networks based on the exact calculation of Pearson's correlation of large time-series. By pre-computing simple and low-overhead statistics, TSUBASA can efficiently compute the exact pairwise correlation of time-series on arbitrary time windows at query time. For real-time data, TSUBASA proposes a fast and incremental way of updating a network at interactive speed. Our experiments show that TSUBASA is faster than approximate solutions at least one order of magnitude for both historical and real-time data and outperforms a baseline for time-series correlation calculation up to two orders of magnitude.

KEYWORDS

time-series, climate data, network analysis, climate networks

ACM Reference Format:

. 2018. TSUBASA: Climate Network Construction on Historical and Real-Time Data. In *SIGMOD '22: ACM Symposium on Neural Gaze Detection, June 12–17, 2022, Philadelphia, PA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

To identify and analyze patterns in global climate, scientists and climate risk analysts model climate data as complex networks – networks with non-trivial topological properties [2, 16, 19]. The climate network architecture represents the global climate system by a set of anomaly time-series (departure from the usual behavior) of gridded climate data and their interactions [34]. A climate data set includes remote and in-situ sensor measurements (e.g. sea surface temperature and sea level pressure) covering a grid (e.g. with a resolution of $2.5° \times 2.5°$). Nodes in a climate network are geographical locations, characterized by time-series and edges represent information flow between nodes. The edge weights indicate a degree of correlation between the behaviors of time-series (e.g. Pearson's correlation). Note the geographical locality of nodes does not directly imply the topology of a network.

Several studies have applied network science on climate data assuming dynamic networks that are changing with real-time data [5].

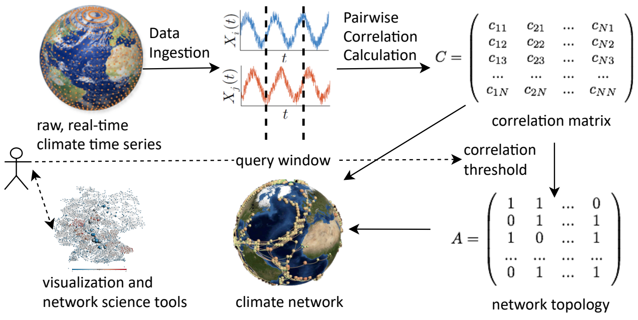

Data Ingestion
$X_i(t)$
$X_j(t)$
t
Pairwise Correlation Calculation
$C = \begin{pmatrix} c_{11} & c_{21} & \dots & c_{N1} \\ c_{12} & c_{22} & \dots & c_{N2} \\ c_{13} & c_{23} & \dots & c_{N3} \\ \dots & \dots & \dots & \dots \\ c_{1N} & c_{2N} & \dots & c_{NN} \end{pmatrix}$
correlation matrix
raw, real-time climate time series
query window
correlation threshold
$A = \begin{pmatrix} 1 & 1 & \dots & 0 \\ 0 & 1 & \dots & 1 \\ 1 & 0 & \dots & 1 \\ \dots & \dots & \dots & \dots \\ 0 & 1 & \dots & 1 \end{pmatrix}$
visualization and network science tools
climate network
network topology

Figure 1: Climate Network Construction.

Climate networks have been shown to be powerful tools for gaining insights on earthquakes [2], rainfalls [19], and global climate events such as El Niño [16].

The common way for network dynamics analysis is to construct networks for each hypothesized time-window and analyze them separately [14]. Figure 1 shows the steps of constructing a climate network. Given a query window provided by a user, a correlation matrix is constructed by computing the pairwise correlation of all time-series on the query window. Pearson's correlation is one of the most dominant measures for studying the pairwise climatical correlation [12]. The correlation matrix enables visualization [28], network dynamics analysis [5], as well as tasks such as community detection [33]. To analyze the topology of the network, a user-provided correlation threshold can be applied on the matrix to find the significant edges between nodes and obtain a boolean network matrix. From the mathematical perspective, the analytical computation of the evolution of a complex system (or even not so complex such as Ordinary Differential Equation systems), depends on the robustness and correctness of the initial weights in the complex network [16].

The core task in network construction is the problem of large-scale all-pair time-series correlation calculation. The key challenges of interactive network analysis include: 1) exact calculation of the complete correlation matrix, 2) correlation calculation on time-windows of arbitrary size, and 3) efficiency of network construction and update for historic and real-time data to achieve interactivity.

The line of data management research that computes networks on time-series (for example, for stock market data or climate data) apply pruning techniques on the approximation of correlation measures. In particular, StatStream [37] and MASS [25] reduce the correlation of time-series to the distance of their Discrete Fourier Transform (DFT) coefficients and propose grid-based indexing [37] and I/O-aware techniques [25] for performing threshold-based correlated time-series search. The accuracy of the network can be increased by considering a very large number of DFT coefficients that are expensive to compute. a query window size is restricted to be an integral multiple of the basic window size, which limits the

SIGMOD '22, June 12–17, 2022, Philadelphia, PA
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

usability of these algorithms. In this paper, we present TSUBASA a framework for efficient construction and update of the correlation matrix for arbitrary query windows on historical and real-time data. The differences of TSUBASA and existing work are three-fold. First, unlike these algorithms that only identify edges with a correlation higher than a threshold, TSUBASA computes and updates the complete correlation matrix. This enables network dynamics analysis as well as choosing arbitrary thresholds at query time. Second, TSUBASA computes the exact correlation between time-series. Finally, unlike the existing work, TSUBASA allows arbitrary query window sizes. In this paper, we make the following contributions.

- We present the mathematical tools for the exact calculation of pairwise Pearson's correlation of time-series using the basic window model.
- TSUBASA relies on simple statistics of basic windows pre-computed by doing one pass over the whole data. This provides a flexible and highly responsive correlation calculation mechanism. Users can obtain a correlation matrix given any query window without computing the correlation statistics repeatedly.
- We propose an incremental solution for real-time update of the correlation matrix and climate network. Relying on the easy-to-compute statistics of basic windows means the correlations can be updated quickly for frequently-updated time-series.
- We enable queries with arbitrary time-window size and start and end point on both historic and real-time by relaxing the restriction of the existing basic window model on a query window size being an integral multiple of basic window size.

2 PRELIMINARIES

We start by introducing the climate network construction problem, then, give an overview of existing approximate techniques for calculating time-series correlation.

2.1 Climate Network

We are given a collection $\mathcal{L} = \{x^1, \ldots, x^n\}$ of geo-labeled time-series, where x^i denotes the time-stamped values of a climatic variable collected at location i. A time-series x^i is defined as $[\mathbf{x}^i_1, \ldots, \mathbf{x}^i_m]$, where $\mathbf{x}^i_j$ is the observed value at time j. We assume all time-series in $\mathcal{L}$ are synchronized, i.e. each time-series has a value available at every periodic time interval, namely time resolution. Particularly, if the time resolution is γ and the current timestamp is j, every x^i is $\mathcal{L}$ will have a value observed at time $j+\gamma$. If an x^i has missing value at j, a value is interpolated or if multiple values appear between j and $j+\gamma$, an aggregate value is assigned. Table 1 shows a list of notations used throughout this paper.

Now, we turn our attention to user's query. At query time, a user defines a query time-window $w = (e, l)$ and a correlation threshold θ. The query window is defined with an end timestamp e and a length l that indicates a sub-sequence of size l in a time-series with a start timestamp $e - l + 1$ and an end timestamp e. For real-time data, the end timestamp can be the last observed time, i.e. a user query on real-time data is $w = (\text{“}now\text{”}, l)$, that means the network is constructed on the last l observed data points. We consider the data points within w for each time-series $x = [\mathbf{x}_1, \ldots, \mathbf{x}_k]$. For example, $[\mathbf{x}_{k-m+1}, \ldots, \mathbf{x}_m]$ is the sequence we consider for x on the query window $w = (k, m)$. When clear from the context, we call the sequence of a time-series x, for a given query window simply query window or time-series x. The climate network of $\mathcal{L}$ for a given query time-window w is a graph $\mathcal{N} = (G, V)$, where a node in G corresponds to a location i and is represented by time-series x^i. An edge in V between nodes i and l indicates that the correlation between time-series x^i and x^l is above a user-defined threshold θ. In this paper, we focus on the most commonly used correlation measure i.e. Pearson's correlation coefficients. Given query windows $x = [\mathbf{x}_1, \ldots, \mathbf{x}_m]$ and $y = [\mathbf{y}_1, \ldots, \mathbf{y}_m]$, with means $\bar{x}$ and $\bar{y}$, the Pearson's correlation of x and y is calculated as follows.

$$Corr(x, y) = \frac{\sum_{i=1}^{m}(\mathbf{x}_i - \bar{x})(\mathbf{y}_i - \bar{y})}{\sqrt{\sum_{i=1}^{m}(\mathbf{x}_i - \bar{x})^2}\sqrt{\sum_{i=1}^{m}(\mathbf{y}_i - \bar{y})^2}} \tag{1}$$

Constructing a network for a query window (e.g. the first six months of 2021) requires computing the correlation for all pairs of time-series and pruning links using threshold θ. Existing techniques for fast all-pair correlation calculation on large time series approximate pairwise correlation by relying on the Fourier transform [11, 25, 37]. In this paper, we present an efficient way of calculating the exact correlation of time-series and constructing a network for historical data and updating the network for real-time data. The existing work divide time-series into cooperative and uncooperative to perform correlation approximation. Although our core contribution is the exact calculation and the update of correlations, we also present an approximate way of calculating correlation for generic time-series (§ 3.2). We start by giving an overview of correlation approximation.

2.2 Correlation Approximation Solutions

Computing the correlation of large-scale time-series is pervasively done using the notion of basic windows [25, 37]. Time-series are processed in batches of size B, i.e. the stream $[\mathbf{x}_1, \ldots, \mathbf{x}_n]$ is equally divided into n/B basic windows, where the j-th basic window contains data $[\mathbf{x}_{(j-1)*B}, \ldots, \mathbf{x}_{j*B}]$. Similarly, a query window is a sequence of basic windows. A query window is assumed to be divisible by the size of basic window. Figure 2 presents a visualization of query and basic windows. The existing techniques *approximate* correlation using the Discrete Fourier Transform (DFT) of basic windows. The DFT of a time-series $x = [\mathbf{x}_1, \ldots, \mathbf{x}_k]$ is a sequence $X = [\mathbf{X}_1, \ldots, \mathbf{X}_k]$ of complex numbers:

$$\mathbf{X}_f = \frac{1}{\sqrt{k}} \sum_{i=1}^{k} \mathbf{x}_i e^{\frac{-j2\pi f i}{k}}, f = 1, \ldots, k \tag{2}$$

where $j = \sqrt{-1}$. Computing DFT coefficients has time complexity $O(n^2)$ in the size of basic window. For normalized time series, DFT preserves the Euclidean distance between two sequences, that is, $Dist(\hat{x}, \hat{y}) = Dist(\hat{X}, \hat{Y})$. The approximation techniques consider the first few DFT coefficients to capture the shape and properties of time-series. It has been shown that the correlation of two time-series can be reduced to the Euclidean distance of the the DFT coefficients of their normalized time-series [32, 37]. The normalization of a basic window $x_i = [\mathbf{x}_1, \ldots, \mathbf{x}_B]$ is $\hat{x_i} = [\frac{\mathbf{x}_1 - \overline{x_i}}{\sigma_i}, \ldots, \frac{\mathbf{x}_B - \overline{x_i}}{\sigma_i}]$, where $\overline{x}_i$ and σ_i are the mean and standard deviation of x_i. The correlation of two time-series can be obtained from the Euclidean distance $d(.,.)$

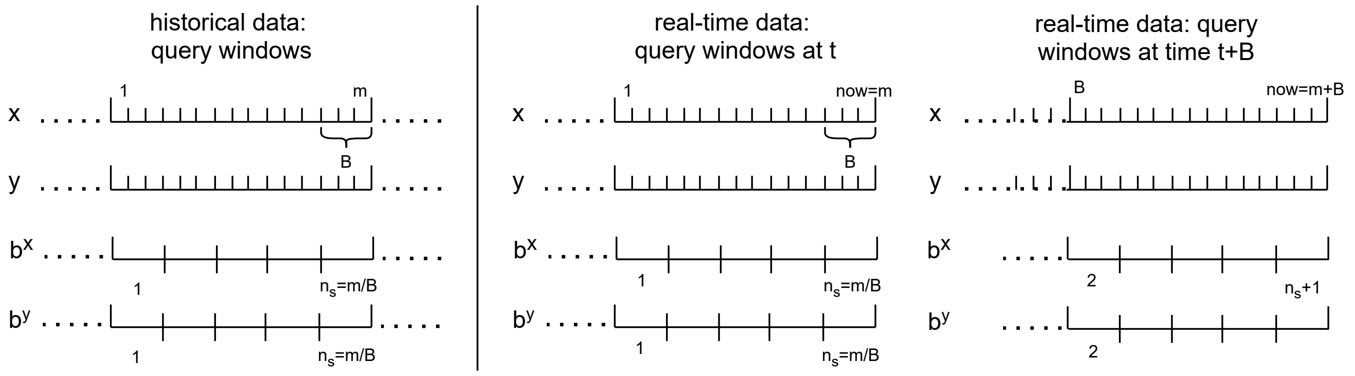

historical data: query windows
real-time data: query windows at t
real-time data: query windows at time t+B
x
y
b^x
b^y
1
m
B
n_s=m/B
now=m
now=m+B
2
n_s+1

Figure 2: Query Windows for Historical and Real-time Data.

of their normalized series.

$$c_i = 1 - \frac{1}{2} d^2(\hat{x_i}, \hat{y_i}) \tag{3}$$

For more concise notation, we denote d_i to be $d(\hat{x_i}, \hat{y_i})$. Suppose $\hat{X_i}$ and $\hat{Y_i}$ are the DFT of normalized basic windows $\hat{x_i}$ and $\hat{y_i}$, and $Dist_n(\hat{X_i}, \hat{Y_i})$ is the Euclidean distance of the first n DFT coefficients in $\hat{X_i}$ and $\hat{Y_i}$. Recall DFT preserves the distance between coefficients and the original time-series. Therefore, $d_i \simeq Dist_n(\hat{X_i}, \hat{Y_i})$. The more coefficients are used (the higher n), the more accurate the distance and correlation becomes. So far, we have a way of computing the distance of basic windows. To compute the distance of query windows, $Dist_n(x, y)$, the existing techniques assume that the form and properties of time-series do not drastically change over a query window, i.e. basic windows have similar statistics (mean and standard deviation) to the query window [25, 37]. When the statistics do not change, $Dist_n(x, y)$ is the average of the d_i on all basic windows of x and y. In § 3.2, we relax this assumption and consider time series that change in form and properties over a query window, i.e. the statistics of basic windows are not necessarily similar to each other and the query window. Now, we apply Equation 3 on time windows, to get $Corr(x, y) \simeq 1 - \frac{1}{2} Dist_n(x, y)$. Again the higher n we use, the better approximation of correlation we obtain.

Now, we describe how $Dist_n(\hat{X}, \hat{Y})$ is used to decide whether $Corr(x, y) \geq \theta$. Zhu and Shasha show the following relationship between correlation and the distance based on n DFT coefficients of $\hat{X}$ and $\hat{Y}$.

$$Corr(x, y) \geq 1 - \epsilon^2 \Rightarrow Dist_n(\hat{X}, \hat{Y}) \leq \epsilon \tag{4}$$

When using approximate techniques for climate network construction, to get the pairs of time-series with $Corr(x, y) \geq \theta$, we can compute $\epsilon = \sqrt{1 - \theta}$. This allows us to prune pairs with condition $Dist_n(\hat{X}, \hat{Y}) \leq \sqrt{1 - \theta}$.

Using Equation 4, we get a superset of highly correlated time-series with no false negatives. As we show in Figure 5a, the false positives incur spurious edges in the network and result in an inaccurate network. These false positive edges can only be filtered at the cost of exact correlation calculation from the raw data. To avoid false positives, TSUBASA calculates exact correlations of time-series, even faster than approximation.

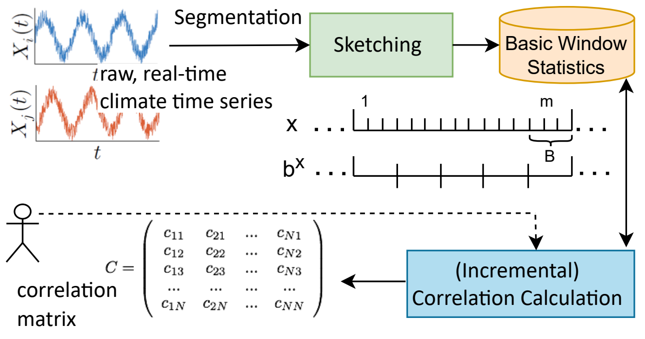
Table 1: Table of Notations

Symbol	Description
x	a query window $[\mathbf{x}_1, \ldots, \mathbf{x}_m]$ of stream $\mathbf{x}$
$\mathbf{x}_i$	data value at time i in a query window x
$\overline{x}$	mean of x
x_j	the j-th basic window of x
$\overline{x_j}$	mean of the j-th basic window of x
$\overline{x_{i:j}}$	mean of basic windows $x_i, \ldots, x_j$ of x
n_s	number of basic windows in a query window
B	number of data points in a basic window
σ_{xj}	standard deviation of the j-th basic window of x
c_j	correlation of x and y on the j-th basic window

Applying Equation 4 requires normalizing time-series and calculating DFT coefficients. When using DFT-based approximation, the accuracy of network increases as more DFT coefficients are considered. Indeed, approximate techniques consider very few coefficients (two in the case of StatStream [37] for any basic window size). However, when dealing with climate data sets, which are uncooperative time-series, the majority of coefficients are needed to get near accurate results (Figure 5a). Statstream proposes random projection for uncooperative time-series that similar to DFT coefficient calculation approximates correlation and has high overhead. To overcome modeling uncooperative time-series, Qiu et al. use Fourier transform and neural network to embed time series into a low-dimensional Euclidean space [31]. The search is done using a nearest neighbor search index in the embedding space. In this paper, we choose a different approach and compute exact correlation and network. Our solution, TSUBASA, relies on simple statistics of basic windows, and as we show empirically is much faster than approximation techniques. In particular, TSUBASA can update a network constructed on real-time data more efficiently than approximation techniques. This enables interactive network analysis on accurate networks for historical and real-time climate data.

3 NETWORK CONSTRUCTION

Before a deep dive into exact correlation calculation, we present a high-level overview of TSUBASA's end-to-end framework. Figure 3 illustrates the components of TSUBASA for constructing and



$$C = \begin{pmatrix} c_{11} & c_{21} & \cdots & c_{N1} \\ c_{12} & c_{22} & \cdots & c_{N2} \\ c_{13} & c_{23} & \cdots & c_{N3} \\ \cdots & \cdots & \cdots & \cdots \\ c_{1N} & c_{2N} & \cdots & c_{NN} \end{pmatrix}$$

Figure 3: Architecture of TSUBASA. update figure with parallel info.

Algorithm 1 PREPROCESSING

```
Input: streams $\mathcal{L} = \{\mathbf{x}^1, \ldots, \mathbf{x}^n\}$; basic window size $B$
Output: statistics $S$
1:  $n_s \leftarrow$ Len($\mathbf{x}^1$)$/B$
2:  $S \leftarrow \{\}$
3:  for $x, y \in \mathcal{L}$ do
4:      $x \leftarrow$ BasicWin($\mathbf{x}, B$); $y \leftarrow$ BasicWin($\mathbf{y}, B$)
5:      for $j \in [1..n_s]$ do
6:          $S_{x_j} \leftarrow$ Stats($x_j$); $S_{y_j} \leftarrow$ Stats($y_j$)
7:          $c_j \leftarrow$ Corr($x_j, y_j$)
8:          $\hat{x}_j \leftarrow$ Normalize($x_j, S_{x_j}$); $\hat{y}_j \leftarrow$ Normalize($y_j, S_{y_j}$)
9:          $\hat{X}_j \leftarrow$ DFT($\hat{x}_j$); $\hat{Y}_j \leftarrow$ DFT($\hat{y}_j$)
10:         $d_j \leftarrow$ Dist$_n$($\hat{X}_j, \hat{Y}_j$)
11:         // 8-10 are performed for approximation method
12:         $S \leftarrow$WriteStats($S_{x_j}, S_{y_j}, c_j, d_j$)
13: return $S$
```

updating a climate network on historical and real-time data. The data storage contains a collection of frequently updated time-series accessible through locations. During the pre-processing, every time-series is divided into basic windows. We sketch basic windows of time-series, in one pass, and store the collected statistics. This can be done at data ingestion time. At query time, the statistics of the basic windows corresponding to a given query window of all time-series are retrieved and all-pair correlation is calculated without the need to access the raw data. For real-time data, the system constructs the initial network and ingests the real-time raw data in chunks of size B. The sketching of the newly ingested basic window is done on the fly and the correlations of time-series are updated incrementally without computing the correlation from scratch. In the following sections, we describe the details of data sketching as well as the mathematical model for the exact and approximate calculation of correlation.

3.1 Exact Pairwise Correlation

3.1.1 Historical Data. Our solution uses the basic window model to calculate the *exact correlation* of times-series. Subdividing a series into basic windows allows us to process data in smaller batches. Existing works for approximate correlation calculation assume

Algorithm 2 NETWORK-CONSTRUCT-HISTO

```
Input: streams $\mathcal{L} = \{\mathbf{x}^1, \ldots, \mathbf{x}^n\}$; statistics $S$; query $w$; basic window size $B$; threshold $\theta$
Output: graph $(G, V)$
1: $G \leftarrow \{1, \ldots, n\}$; $V \leftarrow \{\}$
2: $b \leftarrow$ GetBasicWins($w$) // basic window ids in $w$
3: for $\mathbf{x} \in \mathcal{L}$ and $\mathbf{y} \in \mathcal{L}$ do
4:     $S_x \leftarrow$ ReadStats($S, b, x$); $S_y \leftarrow$ ReadStats($S, b, y$)
5:     $c \leftarrow$ Corr($S_x, S_y$) // use Lemma 1
6:     if $c > \theta$ then
7:         $V$.Add($x, y, c$)
8: return $(G, V)$
```

equal length across all basic windows [25, 37]. Assuming an equal basic window length poses a limitation on the length of the query window, that is the length of the query window could only be an integral multiple of the length of the basic window. Formally, we have $| x | = B \cdot n_s$, where $| x |$ is the length of the basic window. TSUBASA relaxes this assumption and provides a way of calculating the exact pairwise correlation of time-series for arbitrary query window lengths.

LEMMA 1. *Given query windows $x = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m]$ and $y = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m]$ and the sizes of basic windows: $\boldsymbol{B} = [B_1, B_2, \ldots, B_m]$, where B_i is the size of the ith basic window size. The exact Pearson's correlation of x and y is:*

$$Corr(x, y) = \frac{\sum_{j=1}^{n_s} B_j(\sigma_{x_j}\sigma_{y_j}c_j + \delta_{x_j}\delta_{y_j})}{\sqrt{\sum_{i=1}^{n_s} B_i({\sigma_{x_i}}^2 + \delta_{x_i}^2)}\sqrt{\sum_{i=1}^{n_s} B_i({\sigma_{y_i}}^2 + \delta_{y_i}^2)}}$$

$$\delta_{x_i} = \overline{x_i} - \frac{\sum_{k=1}^{n_s} \overline{x_k}}{n_s}, \quad \delta_{y_i} = \overline{y_i} - \frac{\sum_{k=1}^{n_s} \overline{y_k}}{n_s}$$

where, σ_{x_i} (σ_{y_i}) is the standard deviation of basic window of x_i (y_i), c_i is the correlation of basic windows x_i and y_i, $\overline{x_i}$ ($\overline{y_i}$) is the mean of basic window x_i (y_i).

PROOF. This lemma has been provided as a possible general extension provided by Dunlap [13], without proof. We provide a proof here. Let Ω_j be the size of the tail of a time-series with B_1 to B_j basic windows of arbitrary size.

$$\Omega_j = \sum_{k=1}^{j} B_k; \Omega_0 = 0$$

$$
\begin{aligned}
Corr(x,y) &= \frac{1}{T}\sum_{j=1}^{n_s}\sum_{i=\Omega_{j-1}+1}^{\Omega_j}(\frac{\mathbf{x}_i-\overline{x}}{\sigma_x}).(\frac{\mathbf{y}_i-\overline{y}}{\sigma_y})\\
&= \frac{1}{T}\sum_{j=1}^{n_s}\sum_{i=\Omega_{j-1}+1}^{\Omega_j}\frac{\sigma_{x_j}\mathbf{x}^{j,i}+\delta_{x_j}}{\sigma_x}.\frac{\sigma_{y_j}\mathbf{y}^{j,i}+\delta_{y_j}}{\sigma_y}\\
&= \frac{1}{T}\frac{\sum_{j=1}^{n_s}B_j(\sigma_{x_j}\sigma_{y_j}c_j+\delta_{x_j}\delta_{y_j})}{\sigma_x\sigma_y}\\
&= \frac{\sum_{j=1}^{n_s}B_j(\sigma_{x_j}\sigma_{y_j}c_j+\delta_{x_j}\delta_{y_j})}{\sqrt{\sum_{i=1}^{n_s}B_i({\sigma_{x_i}}^2+\delta_{x_i}^2)}\sqrt{\sum_{i=1}^{n_s}B_i({\sigma_{y_i}}^2+\delta_{y_i}^2)}}
\end{aligned}
$$

where $\mathbf{x}^{j,i}(\mathbf{y}^{j,i})$ is the $\mathbf{x}_i(\mathbf{y}_i)$ normalized in the j-th basic window. Now, to show that $\sigma_x = \sqrt{\frac{1}{T}\sum_1^{n_s}B_i({\sigma_{x_i}}^2+\delta_{x_i}^2)}$, we evaluate:

$$
\begin{aligned}
&T{\sigma_x}^2 - \sum_1^{n_s}B_i({\sigma_{x_i}}^2+\delta_{x_i}^2) = \sum_1^{n_s}({\sigma_x}^2-{\sigma_{x_i}}^2-\delta_{x_i}^2)\\
&= \sum_{k=1}^{n_s}\sum_{j=\Omega_{k-1}+1}^{\Omega_k}\frac{1}{B_k}(\mathbf{x}_j-\overline{x})^2-\frac{1}{B_k}(\mathbf{x}_j-\overline{x_k})^2-\frac{1}{B_k}(\overline{x_1}-\overline{x_k})^2\\
&= \sum_{k=1}^{n_s}(\sum_{j=\Omega_{k-1}+1}^{\Omega_k}\frac{(-2\overline{x}\mathbf{x}_j+2\overline{x_k}\,\overline{x})+(2\mathbf{x}_j\overline{x_k}-2\overline{x_k}^2)}{B_k})\\
&= \sum_{k=1}^{n_s}\frac{1}{B_k}((-2B_k\overline{x}\overline{x_k}+2B_k\overline{x_k}\,\overline{x})+(2B_k\overline{x_k}^2-2B_k\overline{x_k}^2)) = 0
\end{aligned}
$$

□

Using Lemma 1, we can pre-compute and store the statistics of basic windows of once and compute the correlation of time-series for user-given time windows at query time without performing a pass over time-series. Moreover, Lemma 1 allows us to support the arbitrary query window lengths. For instance, a user-provided query window $x = [x_i, \ldots, x_j]$ and $y = [y_i, \ldots, y_j]$, there exists a unique $\kappa \in \mathcal{N}$ such that $\kappa \cdot B \le i < (\kappa+1)\cdot B$, and there exists a unique $\chi \in \mathcal{N}$ such that $\chi \cdot B < j \le (\chi+1)\cdot B$. Let $B_1 = (\kappa+1)\cdot B - i$, $B_{n_s} = \chi\cdot B - j$, and B_k, for $k \in \{2, \cdots, n_s-1\}$. At query time, we need to compute $\sigma_{x_1}(\sigma_{y_1})$, $\sigma_{x_{n_s}}(\sigma_{y_{n_s}})$, $\delta_{x_1}(\delta_{y_1})$ and $\delta_{x_{n_s}}(\delta_{y_{n_s}})$ from the raw data, and all the others for the $B_2, \cdots, B_{n_s-1}$ are pre-computed in the pre-processing. Note that the case of equally subdividing time-series into basic windows of size B and a query window size being the integral multiple of the basic window size is a special case of Lemma 1. For this special case, Algorithm 1 shows the steps of sketching basic windows and Algorithm 2 describes the steps of constructing a network based on the exact correlation of time-series calculated from the pre-computed statistics of basic windows.

3.1.2 *Real-time Data.*

The correlation equation of Lemma 1 can be extended to deal with real-time data. A user-defined query window on real-time data, $w = (\text{“}now\text{”}, m)$, indicates the sequence of the m most recently observed data points of time-series. That is, the size of the query window is fixed while the end timestamp is changing as new data arrive. In our problem setting, since the data is processed on the basis of basic windows, the algorithm waits until all new B data points arrive. For time-series $x = [\mathbf{x}_1, \ldots, \mathbf{x}_m]$ and $y = [\mathbf{y}_1, \ldots, \mathbf{y}_m]$, and a query $w = (\text{“}now\text{”}, m)$, we can compute correlation at time t, namely $Corr_t(x,y)$, using Lemma 1. This involves considering basic windows $[x_1, \ldots, x_{n_s}]$ and $[y_1, \ldots, y_{n_s}]$, where $n_s = m/B$. At time $t+B$, the observed time-series are $[\mathbf{x}_1, \ldots, \mathbf{x}_{m+B}]$ and $[\mathbf{y}_1, \ldots, \mathbf{y}_{m+B}]$ and the basic windows are $[x_1, \ldots, x_{n_s+1}]$ and $[y_1, \ldots, y_{n_s+1}]$. Based on query $w = (\text{“}now\text{”}, m)$, we need to consider $[x_2, \ldots, x_{n_s+1}]$ and $[y_2, \ldots, y_{n_s+1}]$. According to Lemma 1, we can recalculate the correlation at time $t+B$ from scratch. That is,

$$
Corr_{t+B}(x,y) = \frac{\sum_{i=2}^{n_s+1}(\sigma_{x_i}\sigma_{y_i}c_i+\delta_{x_i}\delta_{y_i})}{\sqrt{\sum_{i=2}^{n_s+1}({\sigma_{x_i}}^2+\delta_{x_i}^2)}\sqrt{\sum_{i=2}^{n_s+1}({\sigma_{y_i}}^2+\delta_{y_i}^2)}}
$$

Note that δ_{x_j}'s and δ_{y_j}'s have changed and needs to be recalculated, since the means of the new query windows have probably changed upon the arrival of new data. The following lemma allows us to compute $Corr_{t+B}(x,y)$ by only using the statistics of the first and last basic windows, without the need to calculate the statistics of the query window.

Lemma 2. *Given query windows $x = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m]$ and $y = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m]$, basic windows $[x_1, \ldots, x_{n_s}]$ and $[y_1, \ldots, y_{n_s}]$, and basic window size $B = [B_1, \ldots, B_{n_s}]$, where $T = \sum_{i=1}^{n_s}B_i$. Upon the arrival of B_{n_s+1} new data points, we have $x = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{m+B_{n_s+1}}]$ and $y = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{m+B_{n_s+1}}]$ and basic windows $[x_1, \ldots, x_{n_s+1}]$ and $[y_1, \ldots, y_{n_s+1}]$. Let $T' = \sum_{i=2}^{n_s+1}B_i$. Considering a query window $w = (\text{“}now\text{”}, m)$, we can incrementally compute the Pearson's correlation of x and y at time $t + B_{n_s+1}$ from their correlation at time t:*

$$
\begin{aligned}
Corr_{t+B_{n_s+1}}(x,y) = &\frac{1}{C\,.\,D}\Big(T\sigma_x\sigma_y Corr_t(x,y)\\
&+ B_{n_s+1}(\sigma_{x_{n_s+1}}\sigma_{y_{n_s+1}}c_{n_s+1}+\delta_{x_{n_s+1}}\delta_{y_{n_s+1}})\\
&- B_1(\sigma_{x_1}\sigma_{y_1}c_1+\delta_{x_1}\delta_{y_1}) - T'\alpha_x\alpha_y\Big)
\end{aligned}
$$

$$
C = \sqrt{T\sigma_x^2 + B_{n_s+1}({\sigma_{x_{n_s+1}}}^2+\delta_{x_{n_s+1}}^2) - B_1(\sigma_{x_1}^2+\delta_{x_1}^2) - T'\alpha_x^2}
$$

$$
D = \sqrt{T\sigma_y^2 + B_{n_s+1}({\sigma_{y_{n_s+1}}}^2+\delta_{y_{n_s+1}}^2) - B_1(\sigma_{y_1}^2+\delta_{y_1}^2) - T'\alpha_y^2}
$$

$$
\alpha_x = \frac{B_{x_{n_s+1}}\delta_{n_s+1} - B_1\delta_{x_1}}{T} \quad and \quad \alpha_y = \frac{B_{n_s+1}\delta_{y_{n_s+1}} - B_1\delta_{y_1}}{T}
$$

$$
\delta_{x_{n_s+1}} = \overline{x_{n_s+1}} - \overline{x_{1:n_s}} \quad and \quad \delta_{y_{n_s+1}} = \overline{y_{n_s+1}} - \overline{y_{1:n_s}}
$$

where, σ_x (σ_y) is the standard deviation of query window x (y) at time t, σ_{x_j} (σ_{y_j}) is the standard deviation of the basic window of x_j (y_j), c_j is the correlation of the j-th basic windows of x and y, $\overline{x_j}$ ($\overline{y_j}$) is the mean of basic window x_j (y_j), and $\overline{x_{i:j}}$ ($\overline{y_{i:j}}$) is the mean of basic windows $x_i, \ldots, x_j$ ($y_i, \ldots, y_j$).

Proof. We denote $\sigma'_x(\sigma'_y)$ to be the standard deviation of the new query window w after the arrival of the new basic window, and $\delta_{x'_i}(\delta_{y'_i})$ to be parameters of the new query window. We assume there is a linear transform $Corr_{t+B_{n_s+1}}(x,y) = kCorr_t(x,y) + s$, where $k = \frac{T\sigma_x\sigma_y}{T'\sigma'_x\sigma'_y}$ and $s = \frac{1}{T'\sigma'_x\sigma'_y}s'$. Looking at the change of the numerator, we have derived the following for s':

$$\begin{aligned}s' &= T'\sigma'_x\sigma'_y Corr_{t+B_{n_s+1}}(x,y) - T\sigma_x\sigma_y Corr_t(x,y)\\ &= \sum_{i=2}^{n_s+1} B_i(\sigma_{x_i}\sigma_{y_i}c_i + \delta_{x'_i}\delta_{y'_i}) - \sum_{i=1}^{n_s} B_i(\sigma_{x_i}\sigma_{y_i}c_i + \delta_{x_i}\delta_{y_i})\\ &= B_{n_s+1}\sigma_{x_{n_s+1}}\sigma_{y_{n_s+1}}c_{n_s+1} - B_1\sigma_{x_1}\sigma_{y_1}c_1 + \sum_{i=2}^{n_s+1} B_i\delta_{x'_i}\delta_{y'_i}\\ &- \sum_{i=1}^{n_s} B_i\delta_{x_i}\delta_{y_i}\end{aligned}$$

Since $\delta'_{x_i} = \overline{x_i} - (\overline{x_{1:n_s}} + \alpha_x)$ and $\delta'_{y_i} = \overline{y_i} - (\overline{y_{1:n_s}} + \alpha_y)$, we have

$$\begin{aligned}\delta'_{x_i}\delta'_{y_i} &= (\overline{x_i} - (\overline{x_{1:n_s}} + \alpha_x))(\overline{y_i} - (\overline{y_{1:n_s}} + \alpha_y))\\ &= (\delta_{x_i} - \alpha_x)(\delta_{y_i} - \alpha_y)\end{aligned}$$

Then, plugging it into the equation for s', we get

$$\begin{aligned}s' &= B_{n_s+1}(\sigma_{x_{n_s+1}}\sigma_{y_{n_s+1}}c_{n_s+1} + \delta_{x_{n_s+1}}\delta_{y_{n_s+1}})\\ &- B_1(\sigma_{x_1}\sigma_{y_1}c_1 + \delta_{x_1}\delta_{y_1}) - T'\alpha_x\alpha_y\end{aligned}$$

To get s and k, we also need the incremental equation for $\sigma'_x(\sigma'_y)$ if we look at the denominator. In the proof of Lemma 1, we showed that $\sigma_x = \sqrt{\frac{1}{T}\sum_1^{n_s} B_i({\sigma_{x_i}}^2 + \delta^2_{x_i})}$. We have:

$$\begin{aligned}T'\sigma'^2_x &= B_2{\sigma_{x_2}}^2 + B_2\delta'^2_{x_2} + \ldots + B_{n_s+1}\sigma^2_{n_s+1} + B_{n_s+1}\delta'^2_{x_{n_s+1}}\\ &= \sum_{j=1}^{n_s} B_j{\sigma_{x_j}}^2 + B_{n_s+1}{\sigma_{x_{n_s+1}}}^2 - B_1{\sigma_{x_1}}^2 + \sum_{j=2}^{n_s+1} B_j(\delta_{x_j} - \alpha_x)^2\\ &= \sum_{j=1}^{n_s} B_j{\sigma_{x_j}}^2 + B_{n_s+1}{\sigma_{x_{n_s+1}}}^2 - B_1{\sigma_{x_1}}^2\\ &+ \sum_{j=2}^{n_s+1} B_j({\delta_{x_j}}^2 - 2\delta_{x_j}\alpha_x + {\alpha_x}^2)\\ &= T\sigma^2_x + B_{n_s+1}({\sigma_{x_{n_s+1}}}^2 + \delta^2_{x_{n_s+1}}) - B_1(\sigma^2_{x_1} + \delta^2_{x_1}) - T'\alpha^2_x\end{aligned}$$

By replacing k and s in the above transform, the proof of the lemma becomes complete. For real-time data, the mean of the query window changes as new data arrives. For efficiency purposes, we do not want to compute that mean for the calculation of parameters δ. Lemma 1 allows us to compute the standard deviation of a query window and correlation without computing the mean. □

Algorithm 3 describes the steps of constructing a network for real-time data.

3.2 Approximate Pairwise Correlation

This considers non-aribtrary query window size. So far, we presented ways of computing and updating the exact correlation of time-series. Now, we describe how our model can be extended to approximate the correlation of time-series over a query window for all time-series regardless of being cooperativeness or uncooperative. Equation 3 shows how the DFT coefficients of two time series can be reduced to the Euclidean distance of their normalized series, as described in § 2.2 Note that, in our model, the necessary statistics for normalization are collected during the sketch time.

Algorithm 3 NETWORK-CONSTRUCT-REALTIME

```
Input: streams $\mathcal{L} = \{\mathbf{x}^1, \ldots, \mathbf{x}^n\}$; statistics $S$; query $w$; basic window size $B$; threshold $\theta$
Output: graph $(G, V)$
 1: $S \leftarrow Preprocessing(\mathcal{L}, B)$
 2: $G, V \leftarrow Network-Construct-Histo(\mathcal{L}, S, w, B, \theta)$ // create initial network
 3: $b \leftarrow []$ // most recent basic window
 4: while do
 5:     $b \leftarrow$ IngestData()
 6:     if Len(b) == B then
 7:         $s \leftarrow$ Stats($b$)
 8:         UpdateNetwork(G, V, s) // use Lemma 2
 9:         $b \leftarrow []$
10: return
```

3.2.1 *Historical Data.*

Recall d_i is the distance of the normalized i-th basic windows, namely $\hat{x}_i$ and $\hat{y}_i$, $\hat{X}_i$ and $\hat{Y}_i$ are the DFT of normalized basic windows $\hat{x}_i$ and $\hat{y}_i$, and $Dist_n(\hat{X}_i, \hat{Y}_i)$ is the Euclidean distance of the first n DFT coefficients in $\hat{X}_i$ and $\hat{Y}_i$. Since DFT preserves the distance between coefficients and the original time-series, we have $d_i \simeq Dist_n(\hat{X}_i, \hat{Y}_i)$. To compute the distance of query windows, $Dist_n(x, y)$, from the distances of basic windows, without any assumption about the form and properties of basic windows in a query window, we can combine the equation of Lemma 1 and Equation 3 as follows.

$$1 - \frac{1}{2}Dist_n(\hat{X}, \hat{Y})^2 \approx \frac{\sum_{i=1}^{n_s}(\sigma_{x_i}\sigma_{y_i}(1 - \frac{d_i^2}{2}) + \delta_{x_i}\delta_{y_i})}{\sqrt{\sum_{i=1}^{n_s}({\sigma_{x_i}}^2 + \delta^2_{x_i})}\sqrt{\sum_{i=1}^{n_s}({\sigma_{y_i}}^2 + \delta^2_{y_i})}}$$

We simplify the equation and obtain an approximation of the distance of two query windows based on the distances of their basic windows.

$$Dist_n(\hat{X}, \hat{Y})^2 \approx 2 + \frac{\sum_{i=1}^{n_s}\sigma_{x_i}\sigma_{y_i}d_n(\hat{X}_i, \hat{Y}_i)^2 - 2\sum_{i=1}^{n_s}(\sigma_{x_i}\sigma_{y_i} + \delta_{x_i}\delta_{y_i})}{\sqrt{\sum_{i=1}^{n_s}({\sigma_{x_i}}^2 + \delta^2_{x_i})}\sqrt{\sum_{i=1}^{n_s}({\sigma_{y_i}}^2 + \delta^2_{y_i})}} \tag{5}$$

When all DFT coefficients are used, i.e. $n = B$, the $\approx$ becomes $=$, turning into an exact calculation.

To perform all-pair correlation approximation in our framework, we can normalize basic windows and compute their DFT coefficients, and pairwise distances, during the sketch time (lines 8-10 of Algorithm 1). At query time, we use Equation 5 to get $Dist_n(\hat{X}, \hat{Y})$ and apply Equation 3 to obtain the correlation. Algorithm 4 describes the steps of building a network based on the approximation of correlation.

3.2.2 *Real-time Data.*

Combining Equation 5 and Lemma 2, we can get the incremental update equation for approximating pairwise correlation:

Algorithm 4 Network-Approximate

```
Input: streams $\mathcal{L} = \{\mathbf{x}^1, \ldots, \mathbf{x}^n\}$; statistics $S$; query $w$; basic window size $B$; threshold $\theta$
Output: graph $(G, V)$
 1: $G \leftarrow \{1, \ldots, n\}$; $V \leftarrow \{\}$
 2: $b \leftarrow$ GetBasicWins($w$) // basic window ids in $w$
 3: for $x, y \in \mathcal{L}$ do
 4:     $S_x \leftarrow$ ReadStats($S, b, x$); $S_y \leftarrow$ ReadStats($S, b, y$)
 5:     $d_1 \ldots d_{n_s} \leftarrow$ ReadStats($S, x, y$) // distance of basic windows
 6:     if stats of basic windows $\simeq w$ then
 7:         $Dist \leftarrow$ Average($[d_1, \ldots, d_{n_s}]$)
 8:     else
 9:         $Dist \leftarrow$ Distance($S_x, S_y, d_1..d_{n_s}$) // use Equation 5
10:     if $Dist \leq \sqrt{1-\theta}$ then
11:         $V$.Add($x, y, Dist$)
12: return $(G, V)$
```

$$
\begin{aligned}
&2 - Dist_n^{t+B}(\hat{X}, \hat{Y}) \underset{\text{(when n =b)}}{\approx} \\
&\frac{1}{A\,.\,B}\left(n_s\sigma_x\sigma_y Dist_n^t(\hat{X}, \hat{Y}) + \sigma_{x_{n_s+1}}\sigma_{y_{n_s+1}}(1 - \frac{d_{n_s+1})^2}{2}\right) \qquad (6)\\
&- \sigma_{x_1}\sigma_{y_1}(1 - \frac{d_1^2}{2}) - \delta_{x_1}\delta_{y_1} - n_s\alpha_x\alpha_y + \delta_{x_{n_s+1}}\delta_{y_{n_s+1}}\Big)
\end{aligned}
$$

Here, $Dist_n^{t+B}$ ($Dist_n^t(\hat{X}, \hat{Y})$) is the DFT Distance of the query window at time $t + B$ (t) using first n coefficients in each basic window. The new distance can be obtained by calculating the pairwise distances for the last basic window $d_{n_s}+1$.

3.3 Complexity Analysis

In this section, we discuss the complexity analysis of query/sketch time and space overhead of TSUBASA, the DFT-based algorithm, and the baseline algorithm for non-arbitrary query windows. Next, we describe the synergies of time and space with usability. Suppose N is the number of time-series and each time-series is in length L.

Space Complexity The space overhead of TSUBASA is $\psi = \frac{L}{B}(2+\frac{N(N-1)}{2})$, where B is the basic window size and $\frac{L}{B}$ is the number of basic windows since we divide a time-series evenly by default. For each basic window of a time-series, TSUBASA stores two values for the mean and the standard deviation. In addition, for aligned basic windows of all pairs of time-series, TSUBASA stores the correlation of each pair of time-series. As a result, the space complexity of TSUBASA is $O(\frac{LN^2}{B})$. The DFT-based approximate algorithm stores the mean and the standard deviation for basic windows of each time-series and the distance between the first few DFT coefficients of aligned basic windows of pairs of time-series, thus, has the space complexity of $O(\frac{LN^2}{B})$. We remark that this space overhead is in addition to the storage of raw time-series for both algorithms if the raw time-series are not discarded after sketching.

Time complexity The sketch time complexity of TSUBASA is independent of query window size and is $O(L \cdot N^2)$, since TSUBASA requires calculating statistics over the aligned basic windows of all pairs of time-series. The sketch time complexity of the approximate algorithm is worse than TSUBASA and is $O(L^2 \cdot N^2)$, since the calculation of DFT coefficients for a time-series of length L is $O(L^2)$ and coefficients are required for calculating the distance of aligned basic windows in all pairs of time-series. For a query window size $l^* = n_s \cdot B$, both TSUBASA and the approximate algorithm scan all basic windows, therefore, the query time complexity of TSUBASA and the approximate method are both $O(\frac{l^*}{B} \cdot N^2)$. However, the baseline algorithm scans the raw time-series and has the query time complexity of $O(l^* \cdot N^2)$.

The query time complexity of real-time TSUBASA is $O(B^*N^2)$, where B^* is the size of the new coming basic window since TSUBASA needs to compute statistics for the new window. The query time complexity of the real-time approximate algorithm is $O(B^{*2}N^2)$. The query time complexity of real-time baseline algorithm is $O(L^* \cdot N^2)$, where L^* is the size of the query window size.

Usability Discussion Let M be the maximum space capacity available for the storage of time-series sketches. Considering the above space analysis and assuming equal-size basic windows, the minimum basic window size of TSUBASA can be calculated by solving $\frac{L}{B}(2+\frac{N(N-1)}{2}) \leq M$. That is, with M available storage the maximum basic window size handled by TSUBASA is $\frac{L}{M}(2+\frac{N(N-1)}{2})$. Note that both time and space complexity reduce as B increases. Moreover, choosing a large B means less space capacity requirement. Therefore, should we just choose an extremely large B? The answer is no. When an arbitrary query window is not supported, a large B will reduce the flexibility of query windows, thus, usability. For the case of the query window size being the integral multiple of the basic window size, the chosen query window size by users becomes extremely limited. If we consider the generic case of Lemma 1, we will observe a significant rise in query time, since the start/end of a query window can fall anywhere in a basic window, thus, when basic windows are large, the first and last basic windows can be potentially large. Suppose the query window is in length l^*, where $\exists n_s \in R$, such that $n_s \cdot B \leq l^* < (n_s + 1) \cdot B$. The time complexity is $O((\frac{l^*}{B}+B) \cdot N^2)$. When $B > \sqrt{l}$, $\frac{l^*}{B}+B$ is monotonically increasing. Since $B > \sqrt{l}$ at the most meaningful queries, the query time increases when the B increases for the generic method.

3.4 Parallel and Disk-based TSUBASA

The disk-based TSUBASA stores sketches on the disk to be retrieved at query time for correlation calculation. Moreover, despite the quadratic complexity of the sketch time and query time, TSUBASA is embarrassingly parallelizable. The set of all pairs of time-series can be partitioned into groups that are processed in parallel. During sketching, workers are divided into a database worker, that writes statistics to the database, and computation workers, that perform sketch computation. Each worker sketches time-series pairs of a partition and sends the sketches in batches to the database worker to write to a disk-based database. During the query time, each worker is assigned a partition, reads the sketches of time-series in batches directly from the database and computes the pairwise correlations, and outputs a sub-matrix of the correlation matrix.

To leverage data locality and minimize the number I/Os, for partitioning time-series pairs, TSUBASA adopts an approach similar to the parallel block nested loop join. Each partition contains a

Algorithm 5 PRUNING

```
Input: streams $\mathcal{L} = \{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$;threshold $\theta$
Output: Matrix $M(m_{ij})$
 1: $m_{ij} = -\infty$
 2: for $i = 1, 2, \cdots, N$ do
 3:     if $\exists p, q$ s.t. $m_{pq} < 0$ then
 4:         $c_{i1}, \cdots, c_{iN} \leftarrow$ Computecorr($\mathcal{L}, i$)
 5:         for $j = 1, 2, \cdots, N$ do
 6:             for $k = 1, 2, \cdots, N$ do
 7:                 $L_{jk:i}, U_{jk:i} \leftarrow$ Correct-Inference($c_{ij}, c_{ik}$)
 8:                 if $L_{jk:i} \geq \theta$ or $U_{jk:i} \leq -\theta$ then
 9:                     $m_{jk} \leftarrow 1$
10:                 if $L_{jk:i} \geq -\theta$ and $U_{jk:i} \leq \theta$ then
11:                     $m_{jk} \leftarrow 0$
12: $M \leftarrow$ Compute-Rest($M$)
13: return $M$
```

subset of time-series paired with all time-series. i.e. each partition is a group of rows in a correlation matrix and the processing is done row by row in batches. Batches of pairs are assigned to a worker and once a worker is finished, it reads the statistics of the next batch of pairs from the database. Since Pearson's correlation is a symmetric measure, TSUBASA needs to process $n(n-1)/2$ pairs to construct the correlation matrix. For load balancing, TSUBASA assigns the same number of pairs to each worker. Note that the same architecture can be used to make the machinery described, in § 3.2, for correlation approximation.

3.5 Solutions to threshold correlation matrix

Suppose that we know c_{xz} and c_{yz}, and we want to infer the range of c_{xz}. Referring to [?], we have the upper bound and the lower bound of c_{xz}:

$$c_{xz}c_{yz} - \sqrt{(1-c_{xz}^2)(1-c_{yz}^2)} \leq c_{xy} \leq c_{xz}c_{yz} + \sqrt{(1-c_{xz}^2)(1-c_{yz}^2)} \tag{7}$$

The equation above provides us a way in predicting the whole correlation matrix based on a small amount of correlations. For example, we are given N time series and $c_{12}, \cdots, c_{1N}$. We could make prediction on c_{ij} for any $1 \leq i, j \leq N$ from c_{1i} and c_{1j} based on Equation 7. For example, given a positive threshold θ, let $U_{xy:z}$ be the upper bound of c_{xy} estimated from c_{xz} and c_{yz}, and $L_{xy:z}$ be the lower bound of c_{xy} estimated from c_{xz} and c_{yz}. If $L_{xy:z} \geq \theta$ or $U_{xy:z} \leq -\theta$, then we know that $m_{xy} = 1$, where m_{ij} is the i th row and j th column of the output matrix. If $L_{xy:z} \geq -\theta$ and $U_{xy:z} \leq \theta$, then we know that $m_{xy} = 0$.

In the Figure 4, let the horizontal axis be the c_{xz}, and the vertical axis be the c_{yz}. If θ is given to be 0.8, then the colored regions are the cases that we could know the m_{xy} without computing c_{xy}. In the blue regions, $m_{xy} = 1$, and $m_{xy} = 0$ in the red regions. The white region is the uncertainty part, we would need to change a z (anchor) to see whether we could infer/compute further. The algorithm 5 presents how we can use it to prune. We select the anchor randomly (or let every time-series to be the anchor), then if we use the selected anchor to scan the M and see which m_{jk} we could know by the bounds instead of computing c_{jk}.

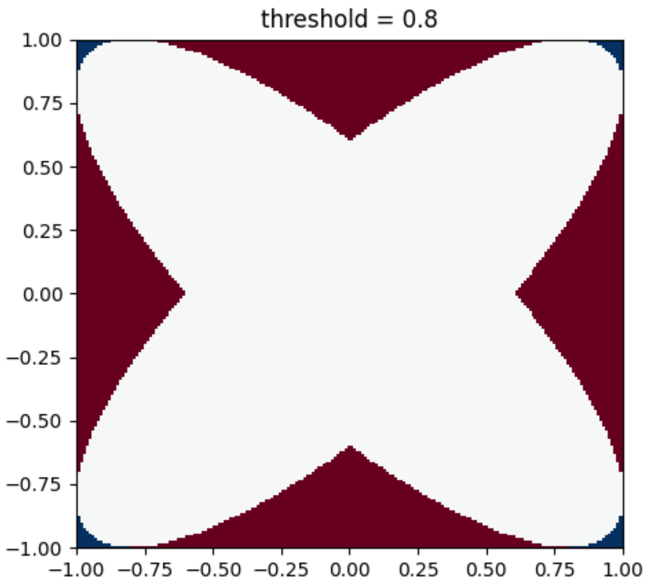


Figure 4: Correlation Inference

4 EXPERIMENTS

We have developed, in this paper, mathematical models and algorithms: NETWORK-CONSTRUCT-HISTO and NETWORK-CONSTRUCT-REALTIME, for constructing and updating correlation matrices to build exact networks on historical and real-time data. Our empirical evaluation has two parts. First, we study these algorithms and compare their query time and sketch time against a baseline, on historical and real-time version of a climate data set. For these experiments, we use the in-memory version of the algorithms, i.e. in-memory data structures are used for storing raw data and sketches. Second, we evaluate the scalability and efficiency of the disk-based and parallel TSUBASA and the approximate algorithm as described in § 3.4. For all experiments, we assume equal basic window sizes in time-series. All algorithms are implemented using Go language. We use PostgreSQL for storing data sketches. All experiments are conducted on a machine with 2 Intel® Xeon Gold 5218 @ 2.30GHz (64 cores), 512 GB DDR4 memory, a Samsung® SSD 983 DCT M.2 (2 TB).

NCEA Data Set[1] is a public data from the National Oceanic and Atmospheric Administration (NOAA). The data is collected every hour, and uploaded publicly in 24-hour increments. NOAA utilizes radiometric satellite collection, buoys, weather stations, citizen scientists, and other methods for perpetual data gathering. The data is collected from 157 nodes (time-series) across the US. Each node produces approximately 8,760 points of data in a year. This data set is used for in-memory experiments.

Berkeley Earth Data Set[2] is a collection of open-source data sets provided by an independent U.S. non-profit organization (Berkeley Earth). We use NetCDF-format gridded data from this data set. The climate data includes average temperature data on both lands

[1]https://www.ncei.noaa.gov/pub/data/uscrn/products/hourly02/2020/
[2]http://berkeleyearth.org/data/

and oceans. It divides the earth by $1^\circ \times 1^\circ$ latitude-longitude grid. We consider the land time-series in this data set. The data set includes 18,638 nodes and each nodes contains 3,652 data points. The time resolution is 24 hours. This data set is used for scalability experiments.

4.1 Accuracy

We compared the accuracy of the climate network of NCEA data set, constructed based on the correlation matrix computed by the DFT-based techniques [25, 37] (as described in § **??**) and exact calculation, followed by the application of a threshold. The approximate technique [11, 37] uses the first few DFT coefficients for estimating the distance of aligned basic windows, then, basic window distances are aggregated to obtain an approximation of the distance and correlation of time-series on a query window [37]. In our experiments, we use the way, we believe, StatStream [37] computes the distance (correlation) of query windows i.e. by averaging the distance (correlation) of DFT coefficients over all basic windows.

We evaluate the impact of approximation on the accuracy of constructed networks, using two measures: number of edges and the correlation similarity ratio, inspired by [26]. A correlation matrix is an $n \times n$ matrix, where n is the number of time-series and a cell c_{ij} is a binary value that indicates the correlation score of time-series x^i and x^j is higher than threshold θ. The correlation similarity ratio evaluates the percentage of identical edges in two networks. Formally, given two complex networks represented by adjacency matrices $A : \{a_{ij} \mid 0 \leqslant i, j \leqslant n\}$ and $B : \{b_{ij} \mid 0 \leqslant i, j \leqslant n\}$, the similarity ratio is defined as follows.

$$D_p(A, B) = \frac{2 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} 1 - |a_{ij} - b_{ij}|}{n(n-1)}$$

For instance, the correlation similarity ratio of networks with the adjacency matrices A and B is $2/3$.

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \quad D_p(A, B) = \frac{2}{3}$$

For both techniques we consider the basic window size 200 and threshold 0.75, while varying the number of DFT coefficients from 50 to 200 for the approximate technique. Note that in the exact technique (basic window correlation), the correlation of time-series are computed by aggregating the correlation of basic windows as suggested by [37]. Therefore, the structure of this network (the solid red plot) is independent of the number of DFT coefficients.

As shown in Figure 5a, the number of edges in the network constructed by DFT correlation calculation becomes equal to the number of edges in the network constructed by exact calculation, only when all 200 coefficients are used. This matches the theory, i.e. the approximation becomes identical to the exact calculation, when all DFT coefficients are used. Note that the approximate technique uses Equation 4 to find correlated time-series based on their DFT-based distance. Following this rule, the DFT correlation calculation never yields false negatives, but, creates false positive edges. This explains why the number of edges in networks decrease as more coefficients are used. Moreover, the similarity ratio of correlation matrices increases as the number of considered coefficients increases and is at its highest value when all coefficients are used to represent a basic window.

The main take-away is that constructing a network based on the approximation of DFT-based distance can lead to inaccurate networks. For climate data sets, near exact result is obtained only when a very large number of coefficients are used for approximation. This means smaller basic windows are preferred for approximation purposes which leads to higher number of basic windows, therefore, higher correlation calculation time in addition to the high DFT coefficient calculation time. These results highlight the necessity of efficient algorithms for constructing and updating exact networks on large collections of time-series.

4.2 Efficiency

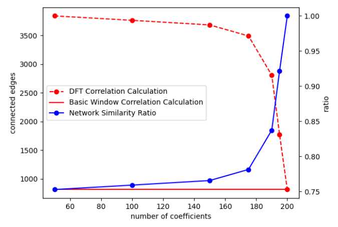
We evaluate the efficiency of the in-memory version of correlation matrix calculation algorithms with respect to query window size and basic window size parameters.

Network Construction We compare the sketch time plus query time when using the DFT-based approximation of StatStream with TSUBASA's exact correlations. For the approximation technique, we report on two scenarios: using all DFT coefficients and using 75% of coefficients of a basic window. As shown in § 4.1, the former empirically yields a network similar to the network of exact correlation calculation. During the sketch time, TSUBASA calculates the statistics from Lemma 1 and the approximation algorithm calculates the statistics necessary for Equation 5 for all basic windows of all time-series. At query time, Lemma 1 and Equation 5 are used to combine sketched statistics to get approximate and exact networks, respectively.

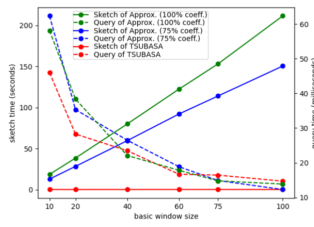
Figure 5b reports the run time when varying the size of basic window for a query window of size 3,000. The sketch time of TSUBASA grows very gradually with the basic window size, while the sketch time of the approximate algorithm increases with the size of basic window. This is because of the $O(n^2)$ complexity of DFT calculation. Our results show that TSUBASA outperforms the approximation technique at sketch time and its query time is on par with the approximate network construction technique.

Figure 5c shows the query time of TSUBASA, approximate calculation, and a baseline, when varying the query window size, considering the constant basic window size of 50. The baseline algorithm computes the Pearson's correlation of Equation 1 for all pairs of time-series directly from raw data at query time without any sketching. In this experiment the approximate algorithm uses 75% of DFT coefficients of a basic window. Note that the distances of basic windows (d_j's in Equation 5) are calculated during the sketch time, therefore, the query time of the approximate algorithm does not depend on the number of considered DFT coefficients. TSUBASA is almost as fast as the approximate algorithm for all query window sizes and outperforms the baseline by two orders of magnitude. We remark that all algorithms have quadratic complexity in the number of time-series. However, the exact and DFT-based approximation are extremely efficient at computing correlation of each pair at query time due to relying on statistics that are pre-calculated during the sketch time.

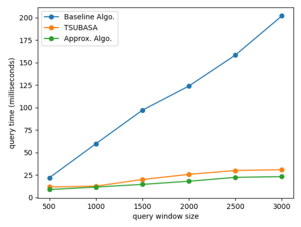
Network Update We compare the network update time of TSUBASA with the DFT-based approximation of § 3.2.2 for real-time



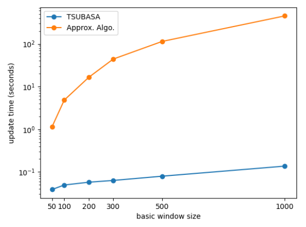
(a)



(b)



(c)



(d)

Figure 5: In-memory: (a) Network Accuracy Comparison (b) Basic Window Size Analysis (c) Query Window Size Analysis (d) Network Update Time.

NCEA data set. The initial networks are constructed on the data set for a given query window. Then, after the arrival of B data points, both algorithms update the correlation and network using the special case of Lemma 2 and Equation 6, respectively. Figure 5d shows the time taken by TSUBASA and approximate algorithm upon the arrival of B new data points for various basic window sizes for a query window size of 3,000. The approximate algorithm uses 75% of DFT coefficients in a basic window. For both algorithms, updates to the network depend on the statistics of the first basic window of the current query window, which is already calculated, and the most recently observed basic window, which needs to be calculated on the fly. Indeed, the efficiency of update only depends on the processing of the most recently observed basic window. Since the approximation algorithm needs to calculate DFT coefficients we observe that it is slower than TSUBASA at least one order of magnitude. The gap between the two algorithms becomes more obvious for larger basic window sizes because of the $O(n^2)$ complexity of DFT calculation. In conclusion, TSUBASA can compute exact correlation and networks for real-time much faster than the approximation competitor.

4.3 Scalability

We compare TSUBASA and the approximation algorithm in similar parallel and disk-based configurations. To separate the impact of fine-tuning the database on performance, in all experiments, we choose to use one database worker and allocate the rest of workers for sketching and querying. For the scalability experiments, we use subsets of time-series from the Berkeley Earth data set. All experiments consider a basic window length of 120, a query window length of 960, and 75% of DFT coefficients for correlation approximation.

Sketch Time Figure 6a shows the sketch time of TSUBASA and the approximate algorithm for correlation matrix calculation for various number of time-series on 63 partitions and 64 cores. The plot separates the write time from the sketch calculation time. We observe that TSUBASA outperforms the approximate algorithm in sketch time. This is due to quadratic complexity of DFT calculation as opposed the linear complexity of computing TSUBASA sketches. We observe that the majority of work by TSUBASA during sketching is spent on writing sketches to a database, unlike matrix approximation which is on par with the write time. Note that in this configuration the total sketch time of TSUBASA and the approximate algorithm is bounded by database write time. The total sketch time, sketch calculation, and write time of TSUBASA and the approximate algorithm increase quadratically with the number of time-series. However, due to parallelization, the growth is slower than what is expected for a single-core configuration.

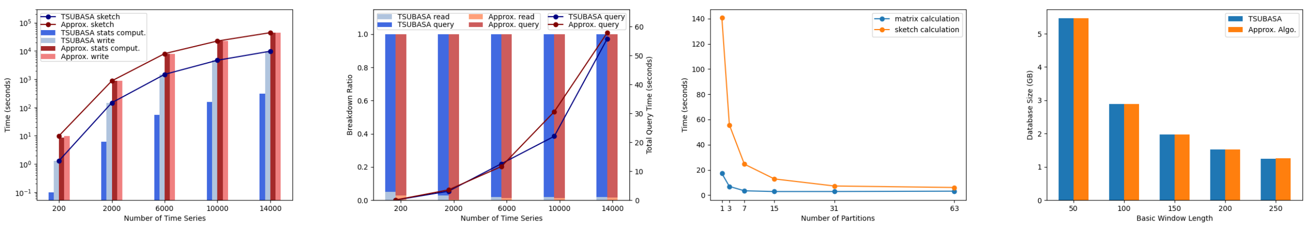
Query Time Figure 6b shows the query time of TSUBASA and the approximate algorithm for correlation matrix calculation for various number of time-series on 63 partitions and 64 cores. The plot separates the database read time from the correlation matrix calculation time. Both TSUBASA and the approximate algorithm have on par query time and take less than a minute for computing the correlation matrix even for the largest number of time-series. We observe that the read time during querying is negligible compared to matrix calculation. The read time percentage is slightly higher for smaller networks due to the database overhead compared to matrix calculation cost on small number of time-series. The total query time, matrix calculation, and read time of TSUBASA and the approximate algorithm increase quadratically with the number of time-series. However, due to parallelization, the growth is slower than what is expected for a single-core configuration.

Impact of Number of Partitions Figure 6c shows how TSUBASA scales with the number of partitions. For these experiments, we use 2,000 time-series. Note than we have 63 partitions/cores for sketch/matrix computation and we reserve one core for database writes. Both sketch and matrix calculation times decrease with the increase in the number of cores. We expect further optimization of query and sketch time can be done by fine-tuning the database and allocating further resources.

Space Overhead Figure 6d shows the size of databases used for storing sketches of 2,000 time-series by TSUBASA and the approximate algorithm with respect to basic window size. Both algorithms store sketches of the the same size for each basic window and have the same space overhead. As the size of basic window increase the number of basic windows decreases and total size of sketches stored by both algorithms decrease.

5 RELATED WORK

Spatio-Temporal Databases represent the value of a climate variable with three dimensions of geometry and time (i.e. latitude, longitude, and timestamp). Systems such as Microsoft StreamInsight [3], GeoMesa [1], and IBM PAIRS Geoscope [20] are designed for processing streams of geospatial data, from sources such as satellites and IoT sensors, benefiting from relational DBMSs, distributed column-oriented databases, and scalable key-value data



(a) **(b)** **(c)** **(d)**

Figure 6: Parallel and Disk-based: (a) Sketch Time Breakdown (b) Query Time Breakdown (c) Impact of Parallelization (d) Space Overhead.

stores. The algorithms and mathematical models we designed in this paper can be incorporated into geo-spatial systems. We use the progressive and declarative processing of Trill [9] for storage and analysis. TSUBASA can be a stand-alone system with other network analysis extensions for clustering and community detection.

Similarity Search on Time-Series To compute the similarity between time-series, several measures have been proposed [24, 30]. In TSUBASA, we consider the Pearson's correlation coefficient as it is the most commonly used measure for building climate networks [14, 34]. There has been extensive work from database community on similarity search of time-series [6, 15, 24, 31, 37]. This line of work considers threshold queries for similarity search in time-series databases and often involve calculating a function over two or more streams and reporting when the threshold is crossed. Time-series similarity search problems in the community have mostly focused on identifying similar time-series to a query time-series. TSUBASA, however, focuses on the construction of the complete and exact correlation matrix, a task that requires all-pair correlation calculation.

Sketching and Data Reduction Alternative techniques to DFT for time-series similarity approximation are Discrete Wavelet Transform (DWT) [10], Singular Value Decomposition [17], and Piecewise Constant Approximation [22]. Data reduction is based on the idea of summarizing the population or sample data through smaller-sized matrices or simple numbers [18]. As for time-series, it has been a topic of interest that reduces data into low-dimensional data while preserving its characteristics to a large extent [36], which has IoT applications [29]. TSUBASA sketches data into statistics that are required for the efficient and exact calculation of correlation scores and networks.

Data Streaming Systems Most data streaming systems, including S4 [27], Muppet [21], Spark [35], and Flink [7] are large-scale data processing systems driven by th Reduce programming model. They can work on different kinds of data coming from real-world sensors or IoT devices. The system we used in the experiments, Trill, also has a distributed version called Quill [8]. The distributed design will definitely reduce the latency of data processing, and also increase the throughput in a given time.

6 DISCUSSION AND CONCLUSION

We presented TSUBASA, an efficient and exact correlation matrix calculation algorithm for climate network construction on historical and real-time data. TSUBASA uses the basic-window model to subdivide a query time-window into smaller windows. TSUBASA computes a cheap and simple sketch of basic windows and reuses them at query time for building networks on arbitrary query windows. We describe a way of approximating time-series correlation and compared it with TSUBASA. Experiments show that TSUBASA can compute exact correlation and network faster than DFT-based approximation techniques. The techniques proposed in TSUBASA can be potentially applied to analyzing stock market data [23] as well as biological data [4]. However, our preliminary investigation shows that approximate and partial calculation of pairwise correlations suffices in scenarios such as stock market data [23]. For future work, we plan to extend our problem definition to unaligned time-series, develop a pairwise correlation pruning algorithm based on a threshold, and consider further optimization of the parallel TSUBASA by fine-tuning.

REFERENCES

[1] [n.d.]. GeoMesa. https://github.com/locationtech/geomesa.

[2] Sumiyoshi Abe and Norikazu Suzuki. 2004. Scale-free network of earthquakes. *EPL (Europhysics Letters)* 65, 4 (2004), 581.

[3] Mohamed H. Ali, Badrish Chandramouli, Balan Sethu Raman, and Ed Katibah. 2010. Spatio-Temporal Stream Processing in Microsoft StreamInsight. *IEEE Data Eng. Bull.* 33, 2 (2010), 69–74.

[4] Albert Batushansky, David Toubiana, and Aaron Fait. 2016. Correlation-based network generation, visualization, and analysis as a powerful tool in biological studies: a case study in cancer cell metabolism. *BioMed research international* 2016 (2016).

[5] Y. Berezin, A. Gozolchiani, O. Guez, and S. Havlin. 2012. Stability of Climate Networks with Time. *Scientific Reports* 2 (2012).

[6] Walter Cai, Philip A. Bernstein, Wentao Wu, and Badrish Chandramouli. 2021. Optimization of Threshold Functions over Streams. *PVLDB* 14, 6 (2021), 878–889.

[7] Paris Carbone, Asterios Katsifodimos, Stephan Ewen, Volker Markl, Seif Haridi, and Kostas Tzoumas. 2015. Apache Flink™: Stream and Batch Processing in a Single Engine. *IEEE Data Eng. Bull.* 38, 4 (2015), 28–38.

[8] Badrish Chandramouli, Raul Castro Fernandez, Jonathan Goldstein, Ahmed Eldawy, and Abdul Quamar. 2016. Quill: Efficient, Transferable, and Rich Analytics at Scale. *PVLDB* 9, 14 (2016), 1623–1634.

[9] B. Chandramouli, J. Goldstein, M. Barnett, and J. F. Terwilliger. 2015. Trill: Engineering a Library for Diverse Analytics. *IEEE Data Eng. Bull.* 38, 4 (2015), 51–60.

[10] Pimwadee Chaovalit, Aryya Gangopadhyay, George Karabatis, and Zhiyuan Chen. 2011. Discrete wavelet transform-based time series analysis and mining. *ACM Comput. Surv.* 43, 2 (2011), 6:1–6:37.

[11] Richard Cole, Dennis E. Shasha, and Xiaojian Zhao. 2005. Fast Window Correlations over Uncooperative Time Series. In *SIGKDD*. 743–749.

[12] Jonathan F Donges, Yong Zou, Norbert Marwan, and Jürgen Kurths. 2009. Complex networks in climate dynamics. *The European Physical Journal Special Topics* 174, 1 (2009), 157–179.

[13] Jack W Dunlap. 1937. Combinative properties of correlation coefficients. *The Journal of Experimental Education* 5, 3 (1937), 286–288.

[14] James H. Faghmous and Vipin Kumar. 2014. A Big Data Guide to Understanding Climate Change: The Case for Theory-Guided Data Science. *Big Data* 2, 3 (2014), 155–163.

[15] Anna Gogolou, Theophanis Tsandilas, Karima Echihabi, Anastasia Bezerianos, and Themis Palpanas. 2020. Data Series Progressive Similarity Search with Probabilistic Quality Guarantees. In *SIGMOD*. 1857–1873.

[16] Avi Gozolchiani, Kazuko Yamasaki, Oz Gazit, and Shlomo Havlin. 2008. Pattern of climate network blinking links follows El Niño events. *EPL (Europhysics Letters)* 83, 2 (2008), 28005.

[17] Jun-Gi Jang, Dongjin Choi, Jinhong Jung, and U Kang. 2018. Zoom-SVD: Fast and Memory Efficient Method for Extracting Key Patterns in an Arbitrary Time Range. In *CIKM*. 1083–1092.

[18] Ravindra Khattree and Dayanand N Naik. 2000. Multivariate data reduction and discrimination. *SAS Institute, Cary, North Carolina* (2000).

[19] Kyunghun Kim, Hongjun Joo, Daegun Han, Soojun Kim, Taewoo Lee, and Hung Soo Kim. 2019. On complex network construction of rain gauge stations considering nonlinearity of observed daily rainfall data. *Water* 11, 8 (2019), 1578.

[20] Levente J Klein, Fernando J Marianno, Conrad M Albrecht, Marcus Freitag, Siyuan Lu, Nigel Hinds, Xiaoyan Shao, Sergio Bermudez Rodriguez, and Hendrik F Hamann. 2015. PAIRS: A scalable geo-spatial data analytics platform. In *Big Data*. 1290–1298.

[21] Wang Lam, Lu Liu, STS Prasad, Anand Rajaraman, Zoheb Vacheri, and AnHai Doan. 2012. Muppet: MapReduce-Style Processing of Fast Data. *Proc. VLDB Endow.* 5, 12 (2012), 1814–1825.

[22] Ge Luo, Ke Yi, Siu-Wing Cheng, Zhenguo Li, Wei Fan, Cheng He, and Yadong Mu. 2015. Piecewise linear approximation of streaming time series data with max-error guarantees. In *ICDE*. 173–184.

[23] Bilal Ahmed Memon and Hongxing Yao. 2019. Structural change and dynamics of Pakistan stock market during crisis: A complex network perspective. *Entropy* 21, 3 (2019), 248.

[24] Katsiaryna Mirylenka, Michele Dallachiesa, and Themis Palpanas. 2017. Data Series Similarity Using Correlation-Aware Measures. In *SSDBM*. 11:1–11:12.

[25] Abdullah Mueen, Suman Nath, and Jie Liu. 2010. Fast approximate correlation for massive time-series data. In *SIGMOD*. 171–182.

[26] Domenico Napoletani and Timothy D Sauer. 2008. Reconstructing the topology of sparsely connected dynamical networks. *Physical Review E* 77, 2 (2008), 026103.

[27] Leonardo Neumeyer, Bruce Robbins, Anish Nair, and Anand Kesari. 2010. S4: Distributed Stream Computing Platform. In *ICDMW 2010, The 10th IEEE International Conference on Data Mining Workshops, Sydney, Australia, 13 December 2010*. IEEE Computer Society, 170–177.

[28] Thomas Nocke, Stefan Buschmann, Jonathan Friedemann Donges, Norbert Marwan, H-J Schulz, and Christian Tominski. 2015. visual analytics of climate networks. *Nonlinear Processes in Geophysics* 22, 5 (2015), 545–570.

[29] Apostolos Papageorgiou, Bin Cheng, and Ernö Kovacs. 2015. Real-time data reduction at the network edge of Internet-of-Things systems. In *2015 11th international conference on network and service management (CNSM)*. IEEE, 284–291.

[30] John Paparrizos, Chunwei Liu, Aaron J. Elmore, and Michael J. Franklin. 2020. Debunking Four Long-Standing Misconceptions of Time-Series Distance Measures. In *SIGMOD*. 1887–1905.

[31] Han Qiu, Hoang Thanh Lam, Francesco Fusco, and Mathieu Sinn. 2018. Learning Correlation Space for Time Series. arXiv:1802.03628 [cs.LG]

[32] Davood Rafiei. 1999. On Similarity-Based Queries for Time Series Data. In *ICDE*. 410–417.

[33] Alexis Tantet and Henk A Dijkstra. 2014. An interaction network perspective on the relation between patterns of sea surface temperature variability and global mean surface temperature. *Earth System Dynamics* 5, 1 (2014), 1–14.

[34] A. A. Tsonis and P. J. Roebber. 2004. The architecture of the climate network. *Physica A* 333 (Feb. 2004), 497–504.

[35] Matei Zaharia, Reynold S. Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J. Franklin, Ali Ghodsi, Joseph Gonzalez, Scott Shenker, and Ion Stoica. 2016. Apache Spark: a unified engine for big data processing. *Commun. ACM* 59, 11 (2016), 56–65.

[36] Xiaojian Zhao. 2006. *High performance algorithms for multiple streaming time series*. New York University.

[37] Yunyue Zhu and Dennis E. Shasha. 2002. StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time. In *VLDB*. 358–369.