Local Correlation Detection with Linearity Enhancement in Streaming Data

Qing Xie[†] Shuo Shang[‡] Bo Yuan[§] Chaoyi Pang[♯] Xiangliang Zhang[†]

[†] Division of CEMSE, King Abdullah University of Science and Technology
[‡] Beijing Key Laboratory of Petroleum Data Mining
Department of Software Engineering, China University of Petroleum-Beijing
[§] Division of Informatics, Graduate School at Shenzhen, Tsinghua University
[♯] Australian E-Health Research Centre, CSIRO
[†]{qing.xie,xiangliang.zhang}@kaust.edu.sa [‡]sshang@cs.aau.dk
[§]yuanb@sz.tsinghua.edu.cn [♯]chaoyi.pang@csiro.au

ABSTRACT

This paper addresses the challenges in detecting the potential correlation between numerical data streams, which facilitates the research of data stream mining and pattern discovery. We focus on local correlation with delay, which may occur in burst at different time in different streams, and last for a limited period. The uncertainty on the correlation occurrence and the time delay make it diff cult to monitor the correlation online. Furthermore, the conventional correlation measure lacks the ability of ref ecting visual linearity, which is more desirable in reality. This paper proposes effective methods to continuously detect the correlation between data streams. Our approach is based on the Discrete Fourier Transform to make rapid cross-correlation calculation with time delay allowed. In addition, we introduce a shape-based similarity measure into the framework, which ref nes the results by representative trend patterns to enhance the signif cance of linearity. The similarity of proposed linear representations can quickly estimate the correlation, and the window sliding strategy in segment level improves the eff ciency for online detection. The empirical study demonstrates the accuracy of our detection approach, as well as more than 30% improvement of eff ciency.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications

Keywords

Data stream; Correlation Detection; Linear Approximation

1. INTRODUCTION

The analysis of correlation between numerical data streams is an important issue for stream mining, and has attracted much scientif c interest. Its major purpose is to explore the interactional link and dependency relationship between data streams, and to a certain degree, evaluate the similarity of streaming data patterns. The occurrence or the loss of correlation usually indicates special events, which help discover groups of objects with similar behaviors or potential anomalies. Due to the important role it plays, correlation has been widely studied for continuous query [10] and sequence pattern mining [15], which benef t the applications of gene expression clustering [20], stock price prediction and physiological analysis.

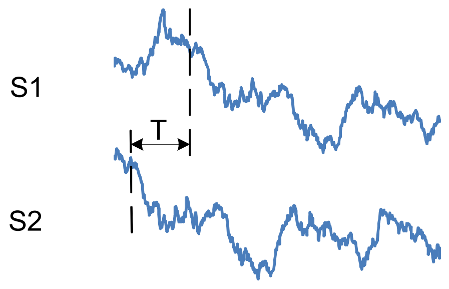


Figure 1: An example of monitoring two streams.

In this paper, we focus on local correlation detection, which may occur in burst, last for certain duration, and then disappear. Unlike the correlation of whole data stream, local correlation is often asynchronous, which is called *lag correlation*. For example, the increase of the exchange rate of Australian dollar may result in the increase of New Zealand dollar, which can be delayed by several hours. Consequently, simply evaluating the similarity of data streams at the same time period can make the correlation inconspicuous. Therefore we propose to take into account the potential time delay for correlation analysis. The property of localization indicates that sliding window is suitable for correlation evaluation and the detection can be performed continuously.

Figure 1 exemplif es the scenario of this problem. We consider monitoring two data streams and capturing the correlated subsequences. It is clear that S_2 has positive correlation with S_1 delayed by time T, but conventional similarity measure methods cannot match them together, since the corresponding subsequences in the same time period are not similar, and the delayed time is unknown in advance. Such situation requires sophisticated approach for correlation detection, which supports analysis based on time delay. However, the uncertainty on the occurrence of correlation and the length of time delay increase the challenge. As the data come continuously, timely reminders should be made if the correlation is identif ed to indicate important events or unexpected cases.

Much effort has been devoted to designing various techniques to continuously match similar subsequences in streaming data [11, 16]. However, correlation represents a more general relationship between streams, which potentially takes place between different kinds of streams. As a result, the conventional distance-based subsequence matching is not suitable for detecting correlation, and instead, more general Pearson coeff cient should be applied. Similar problems have been addressed in [14, 17, 22]. Sakurai et al. [17] designed BRAID to monitor a group of data streams, and detect all the pairs having lag correlation. They proposed to f nd the f rst maximum point on the global cross-correlation coeff cient curve of two data streams, which takes time delay as a variant. However, BRAID considers the whole stream and measures the *global* correlation, which fails to detect *local* correlation. Zhu et al. [22] proposed StatStream based on *Discrete Fourier Transform* to monitor thousands of data streams. Their approach performs well in dealing with large amount of data, but it is designed for ad hoc query other than continuous query, and has strict constraint on the time delay.

Due to the limitations of previous works on this problem, we propose a framework to deal with the continuous detection of local Pearson correlation coeff cient with time delay. Similar to conventional approaches processing streaming data, a sliding window is applied in our framework. Given the maximal time delay allowed, and a minimal correlation duration required, we analyze the subsequences in the sliding window, and f nd if there is any correlation occurring. Since the time delay factor is involved, our solution also employs the *Discrete Fourier Transform* (DFT). Unlike the existing work in [22], we take the advantage of the properties of DFT, and solve the cross-correlation coeff cient with random time shift by reverse DFT on the inner product of two stream sequences. If the correlation is identif ed, incremental evaluation will be performed until the correlation is lost. Otherwise we can slide the window to the next candidate location.

We further address an important issue that the widely applied Pearson coeff cient cannot ref ect the linearity of sequences [1], which delivers more salient meaning in practice. Therefore we propose to ref ne the detection results by linear approximation of the data streams. The linear approximation can describe the data pattern from global and approximated level and help evaluating the linearity by comparing the representations based on the segment similarity. Based on the properties of the approximation technique, we are able to make early pruning to avoid unnecessary calculation by estimating the correlation. In the window sliding mechanism, updating and evaluating the window every time a new data point is received can result in unnecessary comparison and ineff cient sliding. The approximation technique can further optimize the sliding mechanism with eff cient segment-level sliding. The concept of approximation helps dramatically reduce the computational complexity, and make the online detection more practical.

The main contributions of our work are summarized as follows:

- We address the practically important but relatively less studied problem of continuous local correlation detection with time delay. By identifying the scope and limitation of existing works, we combine the concept of lag correlation with continuous local detection, which is a necessary complement for correlation analysis, and can provide more options to explore the relationship among data streams.

- We take the advantage of the properties of DFT to eff ciently evaluate the local cross-correlation of two data streams, which supports random time delay for both subsequences in the sliding window. This approach can effectively identify the cross-correlation with time delay, and in the meantime, achieve high time eff ciency.

- We apply linear representation to approximate the data streams to accelerate the correlation analysis, and design innovative similarity measure methods for two subsequences based on the feature of line segments. It can make early pruning by correlation estimation with proven reliability, and remove the results with less linearity.

The remainder of this paper is organized as follows. In Section 2 we give some basic concepts and the formal problem def nition. We introduce the basic method to calculate the cross-correlation function in Section 3. The linearity enhancement and the sliding mechanism are discussed in Section 4. Experimental results are presented in Section 5, followed by a brief review of related works in Section 6. Finally we conclude our work in Section 7.

2. PRELIMINARIES

First of all, we assume that the numerical data streams are running synchronously and sampled at a f xed rate. Without loss of generality, we take the time interval of sampling as time unit, so that all the sampled data values are continually aligned to consecutive integer scales along the time axis, and the data stream can be treated as a discrete signal sequence. To simplify the problem, we only discuss how to monitor two data streams. It is straightforward to extend the detection to multiple streams. In order to continuously detect the correlation of two streams, we apply sliding window to keep the most recent synchronous subsequences of the streams.

Based on the assumption above, we can formally def ne some fundamental concepts of our problem. The numerical time series streams are denoted as $\tilde{x}$ and $\tilde{y}$, and $x = \{x(0), x(1), \ldots, x(N-1)\}$ stands for the subsequence of $\tilde{x}$ with limited length N, where $x(n) \in \mathbb{R}$ is the value of the $(n+1)^{th}$ data point of the subsequence. We also use x_n for alternative $x(n)$ if no conf ict exists. We def ne $len(x)$ as the length of x, and $st(x)$ is the starting time stamp of x. Similar concepts are also applied to $\tilde{y}$. The problem we propose to address is to continuously discover the local correlation between data streams. Here the correlation is evaluated by widely applied *cross-correlation coeff cient* (Pearson coeff cient). For two sequences x and y, which have the same length N and same starting time stamp, the correlation coeff cient is calculated as:

$$r_{xy} = \frac{\sum_{n=0}^{N-1}(x_n - \bar{x})(y_n - \bar{y})}{\sqrt{\sum_{n=0}^{N-1}(x_n - \bar{x})^2}\sqrt{\sum_{n=0}^{N-1}(y_n - \bar{y})^2}} \tag{1}$$

where $\bar{x}$ and $\bar{y}$ stand for the mean value of x and y respectively.

If time delay is considered, there can be certain time gap between the starting time stamps of two subsequences, between which the cross-correlation is evaluated. Assume the time delay on $\tilde{y}$ is τ and the Equation (1) can be modif ed into:

$$r_{xy}(\tau) = \frac{\sum_{n=0}^{N-\tau-1}(x_n - \bar{x})(y_{n+\tau} - \bar{y})}{\sqrt{\sum_{n=0}^{N-\tau-1}(x_n - \bar{x})^2}\sqrt{\sum_{n=\tau}^{N-1}(y_n - \bar{y})^2}} \tag{2}$$

Here $\bar{x} = \frac{1}{N-\tau}\sum_{n=0}^{N-\tau-1} x_n$ and $\bar{y} = \frac{1}{N-\tau}\sum_{n=\tau}^{N-1} y_n$. Notice that the time delay can also be applied on $\tilde{x}$, and the Equation (1) actually describes $r_{xy}(0)$.

$r_{xy}(\tau) \in [-1, 1]$ indicates the linear correlation of the sequences, and can ref ect the similar behavior of sequence trend. Theoretically, when the evaluated parts of two sequences are correlated, $r_{xy}(\tau)$ will return a value close to 1 or -1. We def ne λ as the time delay giving the maximum cross-correlation coeffcient. That is,

$$\lambda = \arg\max_{\tau} |r_{xy}(\tau)| \tag{3}$$

The correlation score of x and y is given by $cor(x, y) = |r_{xy}(\lambda)|$.

We keep the sliding window over data streams to monitor the most recent data points. When salient correlation occurs between the current subsequences, we detect and report it, as well as corresponding time delay. In order to make the analysis more practical, we require that only those correlated subsequences with long enough duration are meaningful. Also the time delay should be within a tolerant range. Then the problem we are interested in can be formally defned below.

Problem 1 (Local Correlation with Time Delay). *Given two data streams $\tilde{x}$, $\tilde{y}$, when the correlation score of their subsequences $cor(x, y) \geq \delta$, and x, y satisfy: $\lambda \leq T$, $min\{len(x), len(y)\} \geq (L + \lambda)$, report the correlation score and the time delay λ.*

In our work, the maximum time delay allowed T and the minimum duration of the correlation L are given in advance to specify the query. We keep a window with width of $T + L$, so the potential correlation can be explored from current window. During the processing, the data points keep coming, and the sliding window is continuously updated to make dynamic correlation detection.

3. CROSS-CORRELATION EVALUATION

In this section, we will discuss how to evaluate the cross-correlation between different data sequences. We will frst derive a straightforward solution after transforming the Equation (2). We then design the DFT-based approach that can effciently calculate the correlation score to support random time delay in correlation.

3.1 Naive Solution

Let $M = N - \tau$, and we can transform Equation (2) to the following equation:

$$r_{xy}(\tau) = \frac{M \sum x_n y_{n+\tau} - \sum x_n \sum y_n}{\sqrt{M \sum x_n^2 - (\sum x_n)^2} \sqrt{M \sum y_n^2 - (\sum y_n)^2}} \tag{4}$$

The limits of summation are ignored for compact representation. Equation (4) means that the cross-correlation coeffcient can be computed by simple summations of the point values, which support incrementally computation. Based on this equation, we can design a straightforward approach to detect the correlation.

Intuitively, once the sliding window receives the new data points, we incrementally update the basic summations (e.g., $\sum x_n^2$ and $\sum x_n$) of the subsequences, x and y, within the sliding window. Then for every possible time delay $\tau \in [-T, T]$, we calculate the correlation coeffcient according to Equation (4). Here since time delay can be applied on both x and y, when $\tau \leq 0$, $r_{xy}(\tau)$ actually equals $r_{yx}(-\tau)$. When we get the correlation coeffcients for all possible τ values, the maximal value is chosen as correlation score, together with the corresponding τ. If the correlation score is greater than the predefned threshold δ, we report the correlation, and the corresponding τ is returned as time delay λ.

After processing the points in current window, if the correlation condition is satisfed, the current correlated subsequences should be kept, and when the following data points come in, we make the incremental calculation of cross-correlation score until the correlation is lost. Otherwise, the sliding window reads the next data points and recalculate the correlation score.

It is obvious that the naive solution can continuously detect the local correlation between the engaged streams. However, for each new data point and each possible time delay, we have to recalculate the cross-correlation coeffcient, which will result in high computation complexity. The main cost is produced by the sum of inner-product for different τ as described in Equation (4). In the area of signal processing and statistical analysis, the sum of inner-product is usually calculated by DFT for effciency purpose. Therefore, we propose to make use of the theoretical results of DFT to design a more sophisticated approach for correlation detection.

3.2 DFT-Based Solution

In this part, we will frst introduce the basic knowledge of DFT, and provide important lemmas and properties in DFT theory, based on which our DFT-based solution will be elaborated.

Let $x = \{x(0), x(1), \ldots, x(n), \ldots, x(N-1)\}$ be a N-point sequence, and the *Discrete Fourier Transform* of x be $X = \{X(1), X(2), \ldots, X(k), \ldots, X(N-1)\}$, we have

$$X(k) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}kn} \qquad k \in [0, N-1]$$

The inverse Fourier Transform of X is

$$x(n) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X(k) e^{j\frac{2\pi}{N}kn} \qquad n \in [0, N-1]$$

If applying normalization to the sequence x, we can generate the normalized sequence $\hat{x}$, where each sequence point value is

$$\hat{x}(n) = \frac{x(n) - \bar{x}}{\sigma_x} \qquad n \in [0, N-1]$$

Here $\sigma_x = \sqrt{\frac{1}{N} \sum (x(n) - \bar{x})^2}$ is the standard deviation of x.

Similarly, we also normalize the N-point sequence y. By the normalization form of sequence x and y, we can easily educe the following lemma:

Lemma 1. *The correlation coeffcient of x and y can be calculated by the inner-product of normalized sequences $\hat{x}$ and $\hat{y}$,*

$$r_{xy}(\tau) = \sum_{n=0}^{N-\tau-1} \hat{x}(n) \hat{y}(n+\tau).$$

The following theorem describes an important property of DFT, which inspires a new way to calculate the vector inner-product and the cross-correlation of x and y:

Theorem 1. *For two N-point sequences x and y, if $r_{xy}(\tau) = \sum x(n) y(n+\tau)$, then the DFT of $r_{xy}(\tau)$ is given by*

$$R_{xy}(k) = X^*(k) Y(k)$$

where $X(k)$ and $Y(k)$ are the DFT of x and y respectively, and $X^*(k)$ denotes the complex conjugate of $X(k)$.

By Theorem 1[1], the cross-correlation function $r_{xy}(\tau)$ is directly calculated by inverse DFT on R_{xy} after applying DFT on $\hat{x}$ and $\hat{y}$. Since we are able to directly derive the correlation coeffcient with all possible time delay τ, the computation complexity can be signifcantly improved.

For the normalized sequence $\hat{x}$ and $\hat{y}$, we have the following lemma to quickly compute their DFT coeffcients:

Lemma 2. *Let X be the DFT of x, and $\hat{X}$ be the DFT of normalized sequence $\hat{x}$, we have*

$$\hat{X}(k) = \frac{X(k)}{\sigma_x}$$

[1]The proof can be found in [9].

Furthermore, we have

$$\sigma_x = \sqrt{\frac{1}{N}\sum (x_n - \bar{x})^2} = \sqrt{\frac{1}{N}[\sum x_n^2 - \frac{(\sum x_n)^2}{N}]}$$

Hence in order to get the correlation score of x and y, we only need to keep the summations of $\sum x_n^2$, $\sum x_n$, $\sum y_n^2$ and $\sum y_n$, and calculate the DFT of x and y.

However, according to the theory of DFT, the N-point DFT calculation is actually for periodic signals with period of N. When we calculate the DFT of a N-point sequence, the actual effect is to periodically extend the N-point sequence, and return the principal values of the DFT on the periodically extended sequence. So if directly apply the DFT of x and y to calculate the cross-correlation function, error may be brought in by incorrect periodic overlapping when time delay is applied. As a result, we f rst extend x and y with zero values, so that the length of the two sequences increases to $N' \geq 2N - 1$.

By zero extending, we can conclude the following important property:

PROPERTY 1. *The result of cross-correlation function based on inverse DFT on $R_{xy}(k)$ satisf es:*

$$r_{xy}(\tau) = \begin{cases} r_{xy}(\tau) & (0 \leq \tau \leq N-1) \\ r_{yx}(N' - \tau) & (N' - N + 1 \leq \tau \leq N' - 1) \end{cases}$$

It means that $r_{xy}(N' - \tau) = r_{yx}(\tau) = r_{xy}(-\tau)$. So when $0 \leq \tau \leq N - 1$, we can get the correlation coeff cients with time delay on y. At the meantime, we can also get the coeff cients with time delay on x from the $r_{xy}(\tau)$ values when $N' - N + 1 \leq \tau \leq N' - 1$.

As widely proposed in signal processing, we apply *Fast Fourier Transform* to eff ciently calculate the DFT coeff cient. We use a concrete example to test the correctness of the DFT-based vector inner-product calculation. Assume $x = \{1, 2, 3, 4, 5\}$ and $y = \{6, 7, 8, 9, 10\}$, and we f rst extended the original sequences to 16 points for the convenience of FFT calculation. By inverse FFT calculation, the result of vector inner-product function with delay is: $r_{xy} = \{130, 90, 56, 29, 10, 0, 0, 0, 0, 0, 0, 0, 30, 59, 86, 110\}$. From the result we can directly f nd the inner-product value with random time delay, which demonstrates that the approach is working correctly.

Similarly, after processing the points in current window, we can also incrementally keep the detected correlation when receiving new data points. Otherwise we repeat the algorithm to evaluate new data points. The DFT-based approach solves the problem from the view of frequency space, which enables the result to cover time delay on both x and y. As we will discuss in next section, it can also reduce the computational cost.

3.3 Complexity Analysis

In this section, we will discuss the complexity of the naive approach and the DFT-based approach. Since the data stream is potentially inf nite, or semi-inf nite, and the same computation repeats on new arrival points, we only discuss a complete computation for the subsequences in a sliding window. As described in previous sections, we assume the length of sliding window is $N = L + T$, where L is the minimal duration of correlation, and T is the maximal time delay allowed. We can summarize the following conclusions:

LEMMA 3. *For the naive algorithm, the space complexity is $O(N + T)$, and the time complexity is $O(TN)$.*

PROOF. For the naive algorithm, since the maximal time delay is given by T, the correlation should be calculated for $-T \leq \tau \leq T$. Then the number of multiplication operations is

$$N + 2[(N-1) + (N-2) + \cdots + (N-T)] = (T+1)N + (L-1)T.$$

The time complexity is thus $O(TN)$.

The space needed to store the data sequences is $2N$, and the space to store the basic summations is $2(2T + 1)$, so the space complexity is $O(N + T)$. □

LEMMA 4. *For DFT-based algorithm, the space complexity is $O(N)$, and the time complexity is $O(N \log N)$.*

PROOF. The DFT-based algorithm applies FFT to calculate the DFT coeff cients, which involves three complete N'-point FFT calculation, and an N'-point sum product. So the times of multiplication operation is

$$N' + \frac{3}{2}N' \log N' \sim O(N \log N)$$

The space needed to store the data sequences are $2N$, and the storage for FFT calculation is $4N'$ for FFT results and N' for f nal result. So the space complexity for FFT-based algorithm is $5N' \sim O(N)$. □

From the analysis above, we conclude that if the time delay allowed is not large (e.g., $T \ll N$), naive algorithm can perform well. However, in practical applications, the allowed time delay usually varies much, especially when the monitoring window is large. So for the consideration of scalability, we choose DFT-based algorithm as the core calculation algorithm.

4. LINEARITY ENHANCEMENT AND PRUNING TECHNIQUE

In this section, we discuss the utilization of linear representation to enhance the correlation detection, prune the correlation candidates and help the design of eff cient window sliding.

4.1 Linear Approximation

Although to certain degree correlation coeff cient indicates the linear dependency of two sequences, it still cannot precisely characterize the relationship of linearity, which is more desirable and has more semantic meaning in practical applications. As exemplif ed in [1], four pairs of sequences in Figure 2 are in totally different correlationship but with the same correlation coeff cient 0.816. This example indicates that the correlation coeff cient cannot completely identify the linearity and replace the visual examination. Therefore, it is necessary to f nd a representation to describe the visual shape of sequences, which can play the auxiliary role to f lter those sequences dissimilar in shape outlines.

Furthermore, although FFT provides powerful improvement in computation eff ciency, it is still not eff cient enough for online processing, especially when it is required to recalculate the correlation function for each new coming data point. It is a huge waste of time to evaluate every new data point, since there can be much redundancy. To effectively reduce the unnecessary calculation, and make the window sliding more eff cient, we bring in the approximation technique, and propose to estimate the correlation score by certain compact representation.

In our work we apply *Piecewise Linear Representation* (PLR) to approximate the data points. It is to continuously use line segments to approximate the data stream points. As shown in Figure 3, the data stream can be represented by several line segments, which can indicate the trend of the data stream from macroscopic view.

The reasons that we choose linear approximation to represent the data stream fall into three aspects: 1. We are interested with the

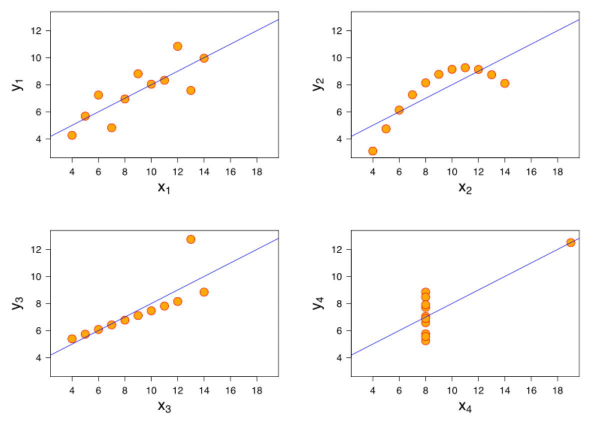


Figure 2: Four pairs of sequences in different correlationships but with the same calculated correlation coefficient.

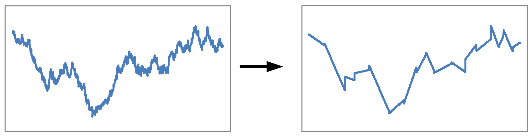


Figure 3: Linear approximation for a data stream.

linear correlation, and the linear representation keeps the overall changing trend of the data stream, so it is the best way to retrieve the global shape feature of a data stream; 2. The compact representation of data stream provides us the trend information to rapidly compare the subsequences and estimate the correlation, which can prune the unnecessary calculation from an early stage; 3. The property of segments inspires us an efficient way to slide the window along data streams. Since the data streams are divided into segments by the linear approximation algorithm, the sliding procedure can go ahead segment by segment rather than point by point, which will significantly reduce the number of subsequence comparisons.

We summarize two criteria for choosing the algorithm to generate PLR: 1. The algorithm is able to control the approximation ratio, so as to make the approximation not too coarse nor too fine; 2. Each line segment minimizes the approximation error on corresponding subsequence, so as to precisely describe the sequence trend. In this work, we apply SWAB algorithm proposed by Keogh et al. [7] to make linear approximation, which can achieve the online efficiency and is suitable for sliding window. We control the PLR generation by the approximation ratio (e.g., the number of segments is fixed). The more details can be found in [7].

4.2 Similarity of Line Segments

As mentioned before, we propose to utilize the compact representation of data stream to estimate the correlation and conduct early pruning. Once the line segments are generated within the sliding window, we can primely evaluate the similarity of the linear representations, which can reflect the linearity similarity between sequences and is capable to filter the subsequences dissimilar in shape.

Since the correlation is sensitive to the increasing and decreasing trend of data point values, we propose to evaluate the similarity of line segments from the view of data changing trend. After the generation of line segments, we propose to transfer the line segments into pairs of numbers, named segment pairs. Each segment pair describes the characteristics of a line segment, which consists of the length of the line segment as well as the changing trend. If the slope of the line segment is greater than 0, we set the changing trend as "1", otherwise the changing trend is "-1". For example, $\{(5,1),(4,-1),(5,1)\}$ means the sequence is approximated by three segments. The first one has 5 points and is increasing; the length of second one is 4, and it is decreasing; the third increasing one has length 5. If duplicate the trend symbols according to the segment length, we can get the following segment trend sequence: $\{1,1,1,1,1,-1,-1,-1,-1,1,1,1,1,1\}$, which reflects the overall changing trend of the data sequence.

So for the two subsequences in the sliding window, we can first compare their similarity by the segment trend sequences. If two sequences are in correlation, their segment trend sequences should be similar, since the correlation indicates the linear dependency of the sequences. We apply *Longest Common Subsequence* (LCS) to evaluate the similarity of two segment trend sequences.

DEFINITION 1 (SEGMENT TREND SEQUENCE SIMILARITY). *Given two segment trend sequences BX and BY with length N, assume BX_i is the i-length prefix of BX, bx_i is the i^{th} element of BX, and $LCS(i,j)$ is the length of the LCS of BX_i and BY_j. Let $sim(BX,BY) = LCS(N,N)/N$ be the similarity score of BX and BY, and it is recursively calculated as follows,*

$$sim(BX,BY) = LCS(N,N)/N$$
$$LCS(i,j) = \begin{cases} 0 & (i=0|j=0) \\ LCS(i-1,j-1)+1 & (bx_i = by_j) \\ max\{LCS(i-1,j), LCS(i,j-1)\} & (bx_i \neq by_j) \end{cases}$$

The Trend Sequence Similarity reflects the similarity of data behavior, or to certain degree, the similarity of stream shape. If the two trend sequences are similar enough according to the Trend Sequence Similarity, we continue on the cross-correlation evaluation as designed in previous sections, otherwise the sliding window moves on to receive new coming points. Since the calculation of LCS only concerns XOR operation other than multiplication, it can be more efficient than the complete correlation analysis, and potentially reduce much unnecessary correlation calculation.

Now we analyze the relationship between the correlation coefficient of two sequences and their Trend Sequence Similarity. The following theorem guarantees that $sim(BX,BY)$ is actually an upper bound of correlation coefficient $r_{xy}(0)$, so that it can safely prune those false candidates.

THEOREM 2. *Given discrete sequences x and y of length N, and their trend sequences BX and BY, the cross-correlation coefficient of x and y satisfies:*

$$r_{xy}(0) \leq sim(BX,BY) = LCS(N,N)/N$$

PROOF. First of all, we have the following inequality:

$$r_{xy}(0) \leq cor(BX,BY)$$

For the two sequences, if certain part has strong correlation, the global trend of the data point should be same. The linear approximation can be viewed as a filtering on the original data stream. Thus after approximation, the noise affecting the correlation is filtered out, and only the principal pattern remains. It is obvious that the correlation coefficient of trend sequences will be larger than that of original streams, and the inequality is established.

Now we prove $cor(BX,BY) \leq sim(BX,BY)$. When BX and BY are the same, assume that for BX, the number of "1" is a, and the number of "-1" is b; for BY, the corresponding numbers are $c = a$ and $d = b$. Then we have,

$$\begin{aligned} cor(BX, BY) &= \frac{N\sum bx_i by_i - \sum bx_i \sum by_i}{\sqrt{N\sum bx_i^2 - (\sum bx_i)^2}\sqrt{N\sum by_i^2 - (\sum by_i)^2}} \\ &= \frac{N^2 - (a-b)(c-d)}{\sqrt{N^2-(a-b)^2}\sqrt{N^2-(c-d)^2}} \\ &= \frac{(a+b)^2-(a-b)^2}{\sqrt{(a+b)^2-(a-b)^2}\sqrt{(a+b)^2-(a-b)^2}} = 1 \end{aligned}$$

Also, $sim(BX, BY)$ satisf es

$$sim(BX, BY) = LCS(N, N)/N = 1 = cor(BX, BY)$$

When BX and BY are different, without loss of generality, we can suppose that $a \geq b$, and for c and d, two cases are possible: $c = a + \varepsilon$, $d = b - \varepsilon$, or $c = a - \varepsilon$, $d = b + \varepsilon$ ($\varepsilon \geq 0$). Assume the Humming distance of BX and BY is ε', ($\varepsilon \leq \varepsilon'$), then the cross-correlation is

$$\begin{aligned} cor(BX, BY) &= \frac{N(N-2\varepsilon') - (a-b)(c-d)}{\sqrt{(a+b)^2-(a-b)^2}\sqrt{(c+d)^2-(c-d)^2}} \\ &= \frac{2ad+2bc-2\varepsilon' N}{4\sqrt{abcd}} \end{aligned}$$

If $c = a + \varepsilon$ and $d = b - \varepsilon$, we have

$$cor(BX, BY) = (ab - a\varepsilon')/(\sqrt{abcd}) \leq (ab - a\varepsilon)/(\sqrt{abcd})$$

If $c = a - \varepsilon$ and $d = b + \varepsilon$, we have

$$cor(BX, BY) = (ab - b\varepsilon')/(\sqrt{abcd}) \leq (ab - b\varepsilon)/(\sqrt{abcd})$$

For the two cases,

$$\begin{aligned} & \frac{(ab-a\varepsilon)}{\sqrt{ab(a+\varepsilon)(b-\varepsilon)}} \leq \frac{(ab-b\varepsilon)}{\sqrt{ab(a-\varepsilon)(b+\varepsilon)}} \\ \Leftrightarrow\ & \frac{a\sqrt{b-\varepsilon}}{\sqrt{a+\varepsilon}} \leq \frac{b\sqrt{a-\varepsilon}}{\sqrt{b+\varepsilon}} \\ \Leftrightarrow\ & a^2(b^2-\varepsilon) \leq b^2(a^2-\varepsilon) \\ \Leftrightarrow\ & b^2 \leq a^2 \end{aligned}$$

It means that the $cor(BX, BY) \leq \frac{(ab-b\varepsilon)}{\sqrt{ab(a-\varepsilon)(b+\varepsilon)}} = \frac{\sqrt{b(a-\varepsilon)}}{\sqrt{a(b+\varepsilon)}}$ always holds.

Now we prove that $\frac{\sqrt{b(a-\varepsilon)}}{\sqrt{a(b+\varepsilon)}} \leq \frac{N-\varepsilon}{N}$ always holds. Assume that $a = rN$ ($0.5 \leq r \leq 1$), then $b = (1-r)N$. The inequality above can be transferred into:

$$\begin{aligned} & \frac{\sqrt{b(a-\varepsilon)}}{\sqrt{a(b+\varepsilon)}} \leq \frac{N-\varepsilon}{N} \\ \Leftrightarrow\ & \frac{\sqrt{(1-r)N(rN-\varepsilon)}}{\sqrt{rN[(1-r)N+\varepsilon]}} \leq \frac{N-\varepsilon}{N} \\ \Leftrightarrow\ & (2r^2-2r+1)N^2 - \varepsilon r(r+1)N + r\varepsilon^2 \geq 0 \end{aligned}$$

For $2r^2 - 2r + 1$, $\Delta_1 = -4 < 0$, so $2r^2 - 2r + 1 > 0$ holds. For the inequality above, $\Delta_2 = \varepsilon^2 r(r^3 - 6r^2 + 9r - 4)$. Let $f(r) = r^3 - 6r^2 + 9r - 4$, the derivative of $f(r)$ is $3r^2 - 12r + 9$, which is larger than 0 within $[0.5, 1]$, so

$$\Delta_2 \leq \varepsilon^2 r f(1) = 0$$

So the objective inequality always holds. □

With the fact that the similarity of segment trend sequences is the upper bound of the cross-correlation coeff cient, trend segment similarity can be used to prune unnecessary correlation calculation. When detecting correlation, cross-correlation coeff cient is measured only if trend sequences suggest potential candidates in current window. Considering the potential time shift, we set the threshold for segment similarity around $L/(L+T)$. (Recall that T is the maximal time delay allowed and L is the minimal duration of the correlation.)

Algorithm 1: segLCS Algorithm

```
   Input: $segBX$,$segBY$
   Output: $segLCS(N_1, N_2)$
1  for $i \leftarrow 1$ to $N_1$ do
2      for $j \leftarrow 1$ to $N_2$ do
3          $L_1 = match(tmpBX_i, segBY_j) + segLCS(i, j-1)$;
4          $L_2 = match(segBX_i, tmpBY_j) + segLCS(i-1, j)$;
5          $L_3 = match(segBX_i, segBY_j, tmpBX_{i-1}, tmpBY_{j-1}) + segLCS(i-1, j-1)$;
6          $segLCS(i, j) = max\{L_1, L_2, L_3\}$;
7          Update $tmpBX_i$, $tmpBY_j$;
8      end
9  end
10 Report $segLCS(N_1, N_2)$;
```

4.3 SegLCS similarity

However, the complexity of LCS calculation is $O(N^2)$, we propose to further improve it based on the characteristics of the trend sequence. As mentioned before, the segment trend sequences can be represented in the form of segment pairs: $\{(5,1), (4,-1), (5,1)\}$, named segment pair sequence. Since the symbols "1" and "-1" both appear as blocks, we can make use of this property and accelerate the calculation of common subsequence. Instead of trend sequences, we directly utilize the block representation of segment pairs, and modify the standard LCS calculation, so that the overall complexity can be dramatically reduced. The new designed algorithm is named segLCS, which is elaborated in Algorithm 1.

We denote the segment pair sequence as $segB$, and the i^{th} segment pair $segB_i = \{len_i, trend_i\}$, where len_i and $trend_i$ are def ned as previously mentioned. During the LCS calculation, we match the whole segments other than each symbol of segments, and record the maximal overlapping part. Notice that after the matching of each pair of segment, we record the remaining parts which are not matched, but could be potentially matched by the following segments. Such remaining part is recorded as $tmpB_i$. For example, given the segment pairs $\{5, 1\}$ and $\{4, 1\}$, the symbol "1" of length 4 are matched, and the remaining part for these two segments are $\{1, 1\}$ and $\{0, 1\}$. So when matching the following segments, the remaining part $\{1, 1\}$ should also be taken into consideration. If the current pairs are $\{5, 1\}$ and $\{4, -1\}$, then the remaining parts should be still $\{5, 1\}$ and $\{4, -1\}$.

Based on the $segLCS(N_1, N_2)$, the similarity between the segment pair sequences is given by $segLCS(N_1, N_2)/N$. The advanced segLCS algorithm can achieve $O(N_1 * N_2)$ complexity, which is a huge improvement on calculation cost. However, segLCS is based on greedy algorithm, which cannot always guarantee the optimal result. So if the requirement for response time is high, we can choose segLCS, otherwise standard LCS can work well.

4.4 Sliding by Segment

The line representation can provide excellent properties to estimate correlation and enhance the linearity. Furthermore, the form of line segment inspires us to scan the stream data according to the line segment. Based on such inspiration, we design an eff cient sliding mechanism to update the sliding window, so as to accelerate the online detection.

As demonstrated in Figure 4, we slide the window according to the endpoints of the line segments in the window. After evaluating

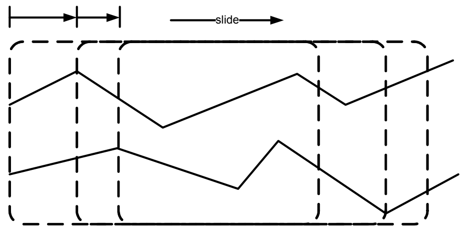


Figure 4: Sliding mechanism based on segment endpoints.

the points in current window, if salient correlation is detected, we can follow incremental evaluation; if we cannot detect salient correlation, the window can be directly moved to the nearest endpoint of the line segments. In this way, the window updating can be more eff cient to reduce much redundant calculation.

5. EXPERIMENTAL EVALUATION

In this part, we describe our experimental evaluation of the proposed framework. The evaluation is conducted on both synthetic and real data, in terms of the effectiveness and eff ciency of the proposed algorithms.

5.1 Experimental Set Up

In our experiments, we perform our algorithm to detect the local correlation between data streams, and compare the results with naive solution that is able to detect all correlation, so as to demonstrate its effectiveness on correlation exploration. Standard recall rate is used for performance measure when f xing the precision rate to be 100%. In order to show the advantage of our algorithm on time eff ciency, we carry it out on data streams with size varying from 10^3 to 10^6, and record the change of time cost. The pruning power brought by linear approximation is also analyzed to show the effective acceleration. All the algorithms are implemented in C++ and all the experiments are performed on a PC with CPU of Intel Core 2.13GHz, and memory of 2 GB.

We test the algorithms on both synthetic and real data streams, so that the conclusion can be made based on a wide range of data distribution. The synthetic data sets are depicted below:

- **Sine Wave**: We use two sequences consisting of sine waves, each of which contains 628 data points. The period of sine wave is 20π, and the amplitude is 10. The sine wave data stream is used to represent classic periodic data.
- **Random**: We also run our algorithm on random data streams, each of the random stream consists of 2000 data points. Random data follow a random-walk model, that is, each data point is lower or higher than the previous data point according to the following relation $y(t+1) = y(t) + \varepsilon$ and $\varepsilon \sim N(0,1)$. The use of random data is for more general correlation occurring at random.

For real data, we choose the following data sets:

- **Sunspots**[2]: The data set describes the daily sunspot numbers from 1/1/1945 to 31/10/2009 with 23680 records. We randomly extract two subsequences from the 23680 records as two data streams, each of which consists of around 1800 data points. The purpose of using this data set is to detect the performance of our algorithm on real periodic data.

[2]`ftp://ftp.ngdc.noaa.gov/STP/SOLAR_DATA/SUNSPOT_NUMBERS/`

Table 1: Comparison on Recall (Precision= 100%)

Data	GT	segLCS		DTW	
		Record	Recall	Record	Recall
Sine	3	3	**100%**	3	**100%**
Random	8	6	75%	1	12.5%
Sunspot	11	8	72.7%	6	54.5%
Temperature	5	5	**100%**	0	0%

- **Temperature**[3]: We use the monthly data records measuring the land temperature and the ocean temperature as data streams. Each of the stream consists of 1560 data points. The correlation between them indicates the link between land temperature and ocean temperature.

5.2 Effectiveness Evaluation

For the effectiveness evaluation, we will test if the proposed framework can successfully detect the potential correlation between data streams, and also compare the effect of our proposed algorithms with conventional technique to measure sequence similarity.

5.2.1 Correlation Detection

We f rst study the performance of correlation detection by our proposed algorithms. Since the naive algorithm can detect all the local correlations, we will f rstly calculate the correlation score at every time stamp using naive solution as ground truth. The correlation score reported by DFT-based algorithm will be further compared with those by naive solution.

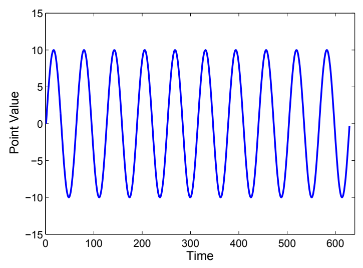
Figure 5 to Figure 8 demonstrate the estimation of our algorithms for all data streams. The blue curve is the correlation score curve calculated by naive solution, and the red triangles record the correlation score of DFT-based algorithm. From all the f gures we can f nd that, the results detected by our algorithms are laid on the correlation score curve at the occurrence of correlation mostly, which demonstrates the effectiveness of our solution. In Figure 5 and Figure 7, the results indicate obviously that the local correlation scores (the red triangles) follow the periodic pattern.

5.2.2 Recall Evaluation

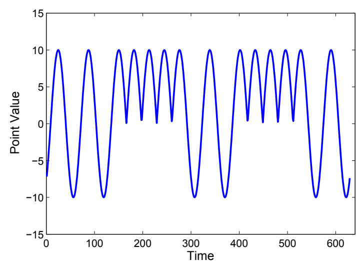
In our approach, we apply linear approximation to improve the linearity of detected results. Hence the algorithm may ignore some subsequences with high correlation score but dissimilar in shape pattern. However, this kind of dismissal is to certain degree subjective to users' visual examination, so it is hard to determine such dismissal is true or false. As a result, we use the precision and recall as evaluation criteria. Since at the seconde phase of the framework, the precision rate can be guaranteed at 100%, we only compare the recall rates. In order to evaluate the effectiveness of our proposed shape-based similarity measure, we also replace segLCS similarity by the classic shape-based similarity measure, Dynamic Time Warping (DTW) distance, to test the corresponding recall rates. The evaluation and comparison results are given in Table 1, where **GT** stands for ground truth of the number of correlated subsequences given by exhaustive search, and **Record** stands for the number of detected correlation by corresponding algorithm.

As summarized in Table 1, the recall rate keeps above 70%, which indicates our algorithm can detect most of the salient correlation with high visual similarity, and effectively reduce the false alarm on the subsequences with low visual similarity. Our results are always superior than that given by DTW distance, and the rea-

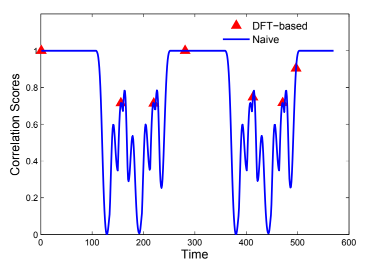
[3]`http://lwf.ncdc.noaa.gov/oa/climate/research/anomalies/index.html`



(a) Sine Stream 1

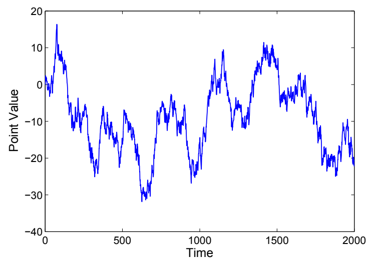


(b) Sine Stream 2

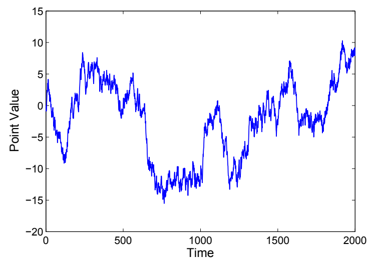


(c) Local Correlation Scores

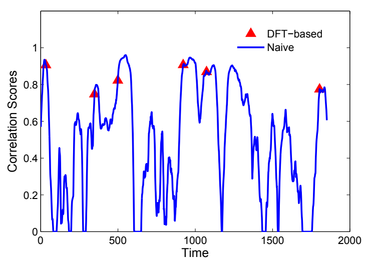
Figure 5: Correlation detection on sine waves: $L = 40, T = 20, \delta = 0.7$



(a) Random Stream 1



(b) Random Stream 2



(c) Local Correlation Scores

Figure 6: Correlation detection on random data: $L = 100, T = 50, \delta = 0.7$

son is our proposed similarity measure can reduce the effect of the value scale of data, and focus on the trend information; while the distance-based approach, e.g., DTW distance, will select those candidates with absolutely high similarity but ignore those with only similar trends, which however, can derive high correlation score. Moreover, it is sensitive to set proper threshold for DTW-based approach to f lter the potential candidates.

5.3 Eff ciency Evaluation

The eff ciency evaluation will focus on the performance of the framework in streaming scenario, and discuss the effect of different factors on the improvement of time eff ciency.

5.3.1 Effect of Data Size

We f rst study the scalability of the algorithms. We compare the running time of naive algorithm, the DFT-based algorithm with standard LCS similarity and that with segLCS similarity on the random data stream. The length of the stream varies from 10^3 to 10^6.

Figure 9 illustrates the changing trend of running time as the data size increases. We can conclude that all algorithms achieve near linear running time to the data size, which is determined by the sliding window framework. However the complexity of DFT-based algorithm is lower than naive solution for processing sliding window, so it can effectively reduce the running time by about 30% over the naive algorithm. We can also explore the advantage of segLCS similarity on time eff ciency, which improves the running time by about 25% over the standard LCS similarity measure.

5.3.2 Effect of Maximal Time Delay

Now we discuss the effect of the maximal time delay on the time eff ciency. As mentioned before, the complexity of naive algorithm is $O(TN)$ (T is the maximal time delay allowed), and $O(N \log N)$

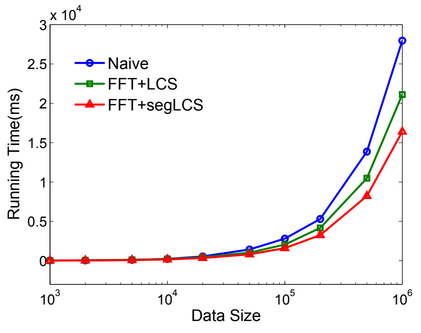
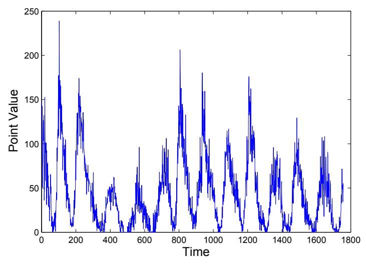


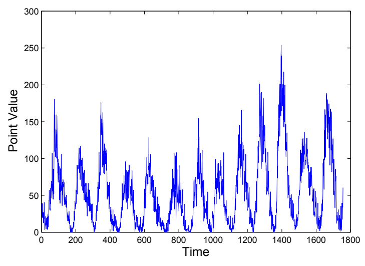
Figure 9: Effect of data size.

for DFT-based algorithm. So the superiority on time eff ciency may be different when the maximal time delay varies. We change the maximal time delay from 50 to 5000, and run both algorithms on the random data with length of 10^5 (the DFT-based algorithm goes without LCS similarity pruning). We compare the running time of both algorithms, and record them in Figure 10.

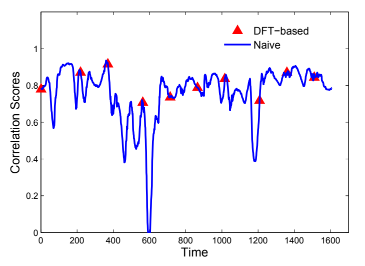
In the f gure, the y-axis stands for the proportion of running time between DFT-based algorithm and naive algorithm. When maximal time delay is small, the running time of DFT-based algorithm is longer than naive solution, because the DFT-based algorithm always calculates the correlation with every possible time delay even if it exceeds T. However, as the growth of T, the DFT-based algorithm outperforms naive algorithm signif cantly, which brings improvement at about 80% off. Such improvement denotes the advantage of FFT calculation on time eff ciency when T is relatively large. Based on the analysis above, we conclude that the choice of algorithm can be adaptive according to the maximal time delay.



(a) Sunspot Stream 1

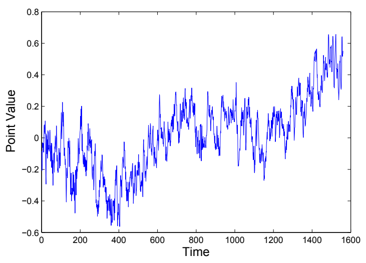


(b) Sunspot Stream 2

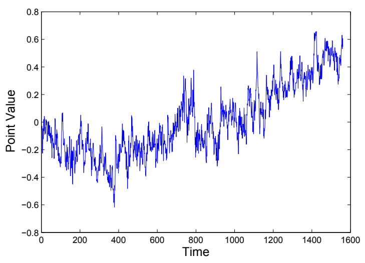


(c) Local Correlation Scores

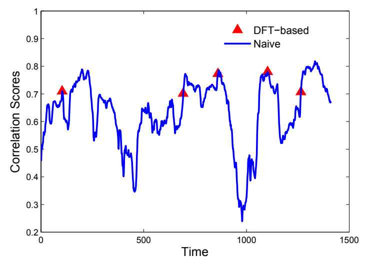
Figure 7: Correlation detection on sunspot numbers: $L = 100, T = 50, \delta = 0.7$



(a) Temperature Stream 1



(b) Temperature Stream 2



(c) Local Correlation Scores

Figure 8: Correlation detection on temperature data: $L = 100, T = 50, \delta = 0.7$

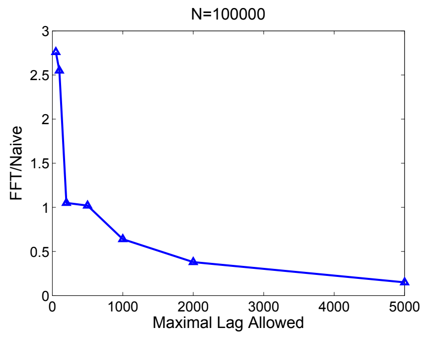


Figure 10: Effect of maximal time delay allowed.

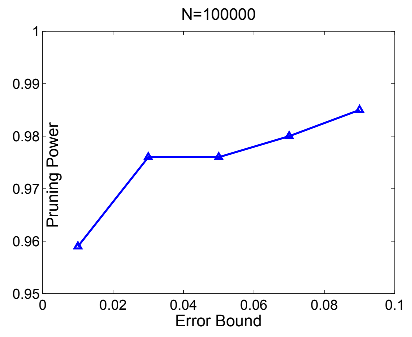


Figure 11: Pruning power analysis.

5.3.3 *Pruning Power Analysis*

Finally, we pay attention to the pruning power brought by line approximation during the sliding mechanism. The pruning power (PP) is defined as:

$$PP = \frac{\text{Number of pruned calculations}}{\text{Number of total correlation calculations}}$$

In this part, we still perform our algorithm on random stream. In order to control the compression ratio, we change the error bound when bottom-up algorithm generates the line segments. As the error bound increases, the compression ratio will also increase, and the linear approximation technique will divide the data stream into less line segments. As a result, when sliding along the data streams, the complete correlation calculation can be reduced dramatically. Figure 11 describes the pruning power brought by linear approximation technique. The curve gives an impressive pruning rate of 95% at least, and when error bound increases, the pruning power can even be improved to 98%.

6. RELATED WORKS

The analysis of time series stream has been engaged for decades. Generally speaking, the techniques in correlation analysis are related to numbers of elementary problems such as time series indexing [8] and continuous queries on data stream.

Indexing technique is a fundamental part of database system, and much work has been done to index time series sequences. Faloutsos et al. [3] applied DFT to represent sequences in database, and map each sequence into points in feature space. Besides the approach in frequency domain, PAA [18] and APCA [6] are proposed to represent and organize time series sequences as line segments in time domain. Indexing techniques can be applied to evaluate the similarity of data sequences, but they usually deal with retrieval in static database, and hard to apply for monitoring streams.

Similar problem is also involved in the research of clustering gene expression data. OP-Cluster [12] and TP-Cluster [13] are designed to discover the positive co-regulation of gene expression on different conditions, that is, the gene expressions synchronously increase or decrease across certain conditions. However, these two algorithms cannot support shifting matching. Zhao et al. [20] proposed Co-GCluster algorithm to successfully f nd both positive and negative co-regulations, but it is not designed for time series data.

Continuous query deals with the problem of evaluating sequence similarity in data stream, which involves similarity measure as well as correlation analysis. *DTW* [2] is applied to continually compare query sequence with the subsequences in data stream in the works of [16, 21]. Lim et al. [11] made continuous sequence matching on data stream supporting variable-length and variable-tolerance. However, continuous query is usually based on distance measure, and rarely considers the relationship of multiple streams.

For stream correlation analysis, Papadimitriou et al. [15] started from the auto-covariance matrices of single data stream, and analyzed the local correlation of two data stream by comparing the eigenvector matrices. Gao et al. [4] addressed the problem of monitoring the streams to discover relevant events/patterns. It applied FFT to eff ciently f nd the cross correlations of time series, and f nd the NN of time series at many time positions. Zhang et al. [19] proposed HBR to transfer time series stream into boolean series successively, so that to reduce the complicated calculation of correlation. However, these works only considered synchronously matching with no time delay.

Zhu et al. [22] proposed a method based on DFT to solve the problem of monitoring tens of thousands of time series data streams in an online fashion and detect the correlation. But there are strict constraints on the supported time delay. BRAID [14, 17] is proposed to deal with the lag correlation issue. It focused on time and space eff ciency during online processing. But since it only calculated the global correlation of data streams, it can hardly applied to solve the local correlation problem. Horvan et al. [5] proposed to use Markov arrival process to approximate the inter-arrival time distribution and the lag correlation. However, it dealt with the traff c process which follows the stochastic modeling.

7. CONCLUSIONS

We address the problem of local correlation detection on two data streams, and allow certain time delay, which is common and signif cant in practical applications. We design a framework to fulf ll the online detection of local correlation, which supports random time delay on random data stream. The core idea of correlation evaluation is DFT-based cross-correlation calculation. Linear approximation is introduced to explore the characteristics of stream shapes and enhance the linearity. The line segment representation is used for global similarity evaluation, which can estimate the correlation score, reduce the unnecessary calculations, and enable the incorporation of an eff cient window sliding approach.

8. REFERENCES

[1] F. Anscombe. Graphs in statistical analysis. *The American Statistician*, 27(1):17–21, 1973.

[2] J. D.J.Berndt. Using dynamic time warping to f nd patterns in time series. In *AAAI workshop on knowledge discovery in databases*, pages 359–370, 1994.

[3] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *SIGMOD'94*, pages 419–429, 1994.

[4] L. Gao and X. S. Wang. Continually evaluating similarity-based pattern queries on a streaming time series. In *SIGMOD'02*, pages 370–381, 2002.

[5] G. Horvath, P. Buchholz, and M. Telek. A map f tting approach with independent approximation of the inter-arrival time distribution and the lag correlation. In *International Conference on the Quantitative Evaluation of Systems*, pages 124 – 133, 2005.

[6] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Locally adaptive dimensionality reduction for indexing large time series databases. In *SIGMOD'01*, pages 151–162, 2001.

[7] E. Keogh, S. Chu, D. Hart, and M. Pazzani. An online algorithm for segmenting time series. In *ICDM'01*, pages 289–296, 2001.

[8] E. Keogh and C. A. Ratanamahatana. Exact indexing of dynamic time warping. *KAIS*, 7:358–386, 2005.

[9] B. Lathi. *Signal processing and linear systems*. Berkeley Cambridge Press, 1998.

[10] H.-S. Lim, J.-G. Lee, M.-J. Lee, K.-Y. Whang, and I.-Y. Song. Continuous query processing in data streams using duality of data and queries. In *SIGMOD'06*, pages 313–324, 2006.

[11] H.-S. Lim, K.-Y. Whang, and Y.-S. Moon. Similar sequence matching supporting variable-length and variable-tolerance continuous queries on time-series data stream. *Information Sciences*, 178:1461–1478, 2008.

[12] J. Liu and W. Wang. Op-cluster: clustering by tendency in high dimensional space. In *ICDM'03*, pages 187 – 194, 2003.

[13] J. Liu, J. Yang, and W. Wang. Biclustering in gene expression data by tendency. In *Proceedings of IEEE Computational Systems Bioinformatics Conference*, pages 182 – 193, 2004.

[14] A. Mueen, S. Nath, and J. Liu. Fast approximate correlation for massive time-series data. In *SIGMOD'10*, pages 171–182, 2010.

[15] S. Papadimitriou, J. Sun, and P. Yu. Local correlation tracking in time series. In *ICDM'06*, pages 456 –465, 2006.

[16] Y. Sakurai, C. Faloutsos, and M. Yamamuro. Stream monitoring under the time warping distance. In *ICDE'07*, pages 1046–1055, 2007.

[17] Y. Sakurai, S. Papadimitriou, and C. Faloutsos. Braid: stream mining through group lag correlations. In *SIGMOD'05*, pages 599–610, 2005.

[18] B.-K. Yi and C. Faloutsos. Fast time sequence indexing for arbitrary lp norms. In *VLDB'00*, pages 385–394, 2000.

[19] T. Zhang, D. Yue, Y. Gu, Y. Wang, and G. Yu. Adaptive correlation analysis in stream time series with sliding windows. *Computers and Mathematics with Applications*, 57:937 – 948, 2009.

[20] Y. Zhao, J. X. Yu, G. Wang, L. Chen, B. Wang, and G. Yu. Maximal subspace coregulated gene clustering. *TKDE*, 20:83–98, 2008.

[21] M. Zhou and M. H. Wong. Eff cient online subsequence searching in data streams under dynamic time warping distance. In *ICDE'08*, pages 686–695, 2008.

[22] Y. Zhu and D. Shasha. Statstream: statistical monitoring of thousands of data streams in real time. In *VLDB'02*, pages 358–369, 2002.