NLC: Search Correlated Window Pairs on Long Time Series

Shuye Pan †, Peng Wang †, Chen Wang ‡, Wei Wang †, Jianmin Wang ‡
† School of Computer Science, Fudan University, Shanghai, China
{pansy20,pengwang5,weiwang1}@fudan.edu.cn
‡ School of Software, Tsinghua University, Beijing, China
{wang_chen,jimwang}@tsinghua.edu.cn

ABSTRACT

Nowadays, many applications, like Internet of Things and Industrial Internet, collect data points from sensors continuously to form long time series. Finding correlation between time series is a fundamental task for many time series mining problems. However, most existing works in this area are either limited in the type of detected relations, like only the linear correlations, or not handling the complex temporal relations, like not considering the unaligned windows or variable window lengths. In this paper, we propose an efficient approach, Non-Linear Correlation search (NLC), to search the correlated window pairs on two long time series. Firstly, we propose two strategies, window shrinking and window extending, to quickly find the high-quality candidates of correlated window pairs. Then, we refine the candidates by a nested one-dimensional search approach. We conduct a systematic empirical study to verify the efficiency and effectiveness of our approach over both synthetic and real-world datasets.

PVLDB Reference Format:
Shuye Pan †, Peng Wang †, Chen Wang ‡, Wei Wang †, Jianmin Wang ‡ .
NLC: Search Correlated Window Pairs on Long Time Series. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at http://vldb.org/pvldb/format_vol14.html.

1 INTRODUCTION

Due to the increasing of data collection frequency of sensors and the rapid progress in distributed storage systems, many up-to-date applications produce a large number of long time series [13]. For example, a sensor with frequency 1Hz can generate a time series longer than thirty million data points in one year ($3600 * 24 * 365 = 3.15 * 10^7$).

When analyzing time series data, an essential task is to assess the correlation between time series. For example, correlation can be used as an intuitive and handy similarity measure in searching, categorizing, classifying and clustering time series [4]. However, when analyzing long time series, measuring correlations between time series becomes subtle. The well known and popularly used global measures, such as Pearson coefficient, are not effective, since the chance that two time series are consistently correlated over a long period is very small and unrealistic. More often than not, two time series may be correlated on some intervals of time, but not the whole time period [16, 18].

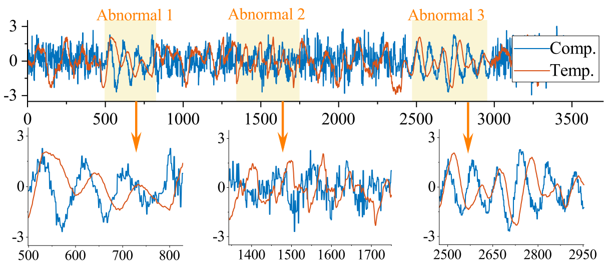


Figure 1: Example of TE benchmark

There also exist some works of local correlation mining, whose goal is to find the subsequence on which two time series are correlated. Jocor [18] mines the correlated subsequence pair from two time series which satisfies 1) the subsequence length exceeds a threshold, 2) the correlation coefficient is maximized. DCI [16] queries the longest correlated subsequence on an aligned time series set given a query. Matrix Profile [27] aims to efficiently compute the correlation between all possible fixed-length subsequence pairs of two time series. Wang et. al. [25] proposes an approach to mine local linear relationships on gapped subsequences.

All these works aim to mine local *linear* correlations from the long time series pair. However, in many applications, the time series have strong non-linear relationships due to the complex physical or chemical mechanism [7, 26]. Below are two typical examples.

- In the industrial domain, Prognostics and Health Management (PHM) is a crucial technology used to maintain the reliability of engineering equipment and systems. In PHM applications, a fundamental task is to find the correlations between monitored signals. The signals are often nonlinearly correlated due to reasons like kinetic relationships, external disturbances, and feedback control, etc [17, 26].
- In the medical and biology domain, multi-dimensional time series classification is used widely [7]. To improve the accuracy, researchers utilize some non-linear measurements, like Granger causality and mutual information, to measure the relevance of variable time series to reduce the redundancy.

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

We illustrate it with a concrete example. Tennessee eastman (TE) process benchmark [6] is a popular realistic simulation model of an industrial chemical process. Consider a pair of time series, one is the temperature and the other is the component, which are collected during variable working conditions, as shown in Figure 1. The Pearson correlation between the whole time series is 0.02. However, when the system is working during conditions “abnormal 1”, “abnormal 2” and “abnormal 3”, these two time series have certain non-linear correlation. The corresponding Pearson coefficients are -0.13, 0.05 and -0.03 respectively. Moreover, we cannot recognize them with DTW distance either, because the component fluctuates more frequently while the temperature fluctuates smoothly. It is interesting that in all three correlations, the mutual information between these two time series exceeds 0.8, which means that they are non-linearly correlated during these three periods.

In this paper, our goal is to search correlated time window pairs from two long time series. We utilize the mutual information as the correlation measurement due to its popularity. Mutual information (MI for short) is a measure of the mutual dependence between two random variables in information theory. It is widely used in variable domains to discover various types of correlation relations, including linear and non-linear, monotonic and non-monotonic, functional and non-functional [9].

Although the mutual information is already utilized in many applications, there exists few works to study how to efficiently search the window pairs with high mutual information values. It is a challenging problem due to the following reasons. First, computing MI is much more costly than the traditional measurement, like Pearson correlation. Because it needs to estimate the density function of each point. Second, the correlation may occur in variable-length windows with variable time delay, which increases the search space greatly. Third, it has not the monotonic property, which hinders us from using the traditional pruning strategy.

Recently, Nguyen Ho et. al. proposes TYCOS to the time delay correlation search problem [9, 10]. It first generates a set of window pair candidates, and then uses the Hill Climbing based approach to find the final correlated pairs. However, TYCOS suffers from two drawbacks. First, during the candidate generation phase, it only searches the aligned correlated window pairs. Therefore, it works poor to find the window pairs with large time delay. Second, the hill climbing approach makes it prone to find *local* optimal results, instead of the *true* ones.

To solve this problem, in this paper, we propose a two-phase approach, Non-Linear Correlation search (NLC for short). In the first phase, we generate a set of correlated window pair candidates. Two strategies, window shrinking and window extending, are proposed. The window shrinking strategy is more efficient which works for the sparsely distributed correlations. It can quickly prune the unqualified intervals. In contrast, the window extending strategy is slightly slower, but it can find more pairs when the correlation windows are distributed more densely. In the second phase, we treat the candidate refinement as an optimization problem and solve it with a nested one-dimension search strategy.

In summary, the contributions of this paper are as follow:

- We carefully analyze three factors influencing the efficiency of the mutual information search, and present the motivation for our approach.
- We propose a two-phase approach, NLC. In the first phase, two strategies are proposed to generate the candidate set. Then in the second phase, we present a nested one-dimension search algorithm to refine the candidates.
- We conduct extensive experiments to verify the effectiveness and efficiency of the proposed approach on two synthetic datasets and three real-world datasets.

The rest of the paper is organized as follows. In Section 2, we present the problem statement and analyze the challenges. In Section 3, we motivate our approach with illustrative examples and overview our algorithm. In Section 4 and Section 5, we introduce the two phases of our approach in detail. The experimental results are presented in Section 6 and related works are discussed in Section 7. The paper is concluded in Section 8.

2 PRELIMINARY

Definition 1 (Time series). *A time series $X = \{x_1, x_2, \cdots, x_n\}$ is a sequence of data arranged in chronological order, where $n = |X|$ is the length of X.*

Definition 2 (Time window). *A time window $W = (s, l)$ is a continuous subsection of the entire time period ($1 \leq s \leq n - l + 1$). The projection of series X on window W, denoted as X_W,or $X(s, l)$, is a subsequence with the starting point s and the length l, i.e., $X_w = \{x_s, ..., x_{s+l-1}\}$.*

Later in the paper, we call W and X_W as the time window interchangeably.

Definition 3 (Pair of Time series). *A pair of time series $(X, Y) = (\{x_1, x_2, \cdots, x_n\}, \{y_1, y_2, \cdots, y_n\})$ consists of data collected from two time series X and Y during the same observation period, and the length of the time series pair is $n = |(X, Y)|$.*

Definition 4 (Pair of time windows). *If two time windows, $X(s, l)$ and $Y(s', l)$, are equal in length, we can form a pair of time windows $(X(s, l), Y(s', l))$. The starting points s and s' may be different, and the time delay is defined as $\tau = s' - s$. More conveniently, the time window pair can also be represented as a triplet $< s, l, \tau >$ by using the starting point s of X, the length l, and the time delay τ.*

Definition 5 (Mutual information of a time window pair). *Mutual information (MI) is a measure of the mutual dependence between two random variables in information theory. Given two discrete random variables X and Y, MI is calculated as follows:*

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x, y) log \frac{p(x, y)}{p(x)p(y)} \quad (1)$$

where $p(x)$ and $p(y)$ are the marginal probability mass functions of X and Y respectively, and $p(x, y)$ is the joint probability mass function. The larger the MI value is, the more correlated the two variables are. $I(X;Y)$ is zero if and only if X and Y are independent.

Definition 6 (Correlated time window pair). *For a time window pair, $X(s, l)$ and $Y(s', l)$, if $I(X(s, l), Y(s', l)) \geq \theta$, where θ is the threshold, we say $X(s, l)$ and $Y(s', l)$ are a correlated time window pair. It can also be called as correlated pair and denoted by the triplet $CP =< s, l, \tau >$, where $\tau = s' - s$ is the time delay. Two correlated pairs $CP_i =< s_i, l_i, \tau_i >$ and $CP_j =< s_j, l_j, \tau_j >$ (assuming $s_i < s_j$)*

are disjoint, if it holds that $s_i + l_i \le s_j$ and $s_i + \tau_i + l_i \le s_j + \tau_j$. Otherwise, the two pairs are overlapping.

Definition 7 (Directly reachable time window pair). *We call two correlated pairs CP_i and CP_j are directly reachable, if there exists a correlated pair CP_k satisfying: 1) CP_i and CP_k are overlapping; 2) CP_j and CP_k are overlapping. As a special case, two overlapping pairs are directly reachable.*

Definition 8 (Significant time window pair). *Correlated pair CP_i is significant if it holds that $I(CP_i) \ge I(CP_j)$ for any correlated pair CP_j direct reachable with CP_i.*

Since in real applications, the correlation will last for a reasonable time period, we constrain the window length neither too long nor too short. Moreover, if two windows, one from X and the other from Y, are correlated, they should not be far away. So, we also constrain the time delay. These two constraints can be determined by the problem scenario and expert knowledge.

Problem statement: Given a pair of long time series (X, Y), a threshold θ, the length constraint $[L_{min}, L_{max}]$ and the delay constraint $[\tau_{min}, \tau_{max}]$, our goal is to find a set of disjoint significant time window pairs, $\mathcal{CP} = \{CP_1, CP_2, \cdots, CP_{|\mathcal{CP}|}\}$, so that $|\mathcal{CP}|$ is maximized. Each $CP_i =< s_i, l_i, \tau_i >$ is a correlated pair, $X(s_i, l_i)$ and $Y(s_i + \tau_i, l_i)$, which satisfies

$$I(X(s_i, l_i), Y(s_i + \tau_i, l_i)) \ge \theta$$

$$l_i \in [L_{min}, L_{max}] \text{ and } \tau_i \in [\tau_{min}, \tau_{max}]$$

As an extreme case, it may happen that two significant pairs are overlapping and have exact MI values. In this case, we randomly select one of them.

2.1 MI Computation

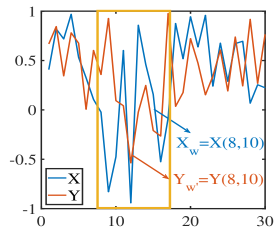
Eq. 1 is the theoretical definition of MI but is usually not used for calculation, as it needs the distributions of the underlying data which are often unknown. In practice, analysts use estimators.

KSG estimator [15] is a popular MI estimator, which simulates the density function by k nearest neighbors. Formally, given window pair, $X_W = X(s, l)$ and $Y_{W'} = Y(s', l)$, the MI value can be efficiently estimated as:

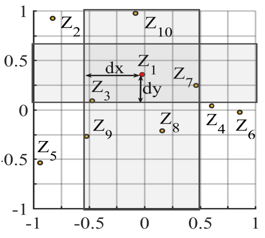
$$I(X_w, Y_{w'}) = \psi(k) - \frac{1}{k} - \frac{1}{l} \sum_{z_i \in (X_w, Y'_w)} [\psi(n_x(z_i)) + \psi(n_y(z_i))] + \psi(l) \tag{2}$$

where ψ denotes the digamma function. Data points in X_W and $Y_{W'}$ correspond one-to-one, which can form two-dimensional data points, *i.e.*, $z_i = (x_{s+i-1}, y_{s'+i-1}), 1 \le i \le l$. For each point z_i, KSG estimator rank neighbors' distances z_i by Chebyshev distance $d_{i,j} = ||z_i - z_j||$, to obtain $d_{i,j_1} \le d_{i,j_2} \le ... \le d_{i,j_{l-1}}$. Then it gets the k nearest neighbors of z_i. dx and dy represent the maximum length in the X and Y directions between the k points and z_i. Then we count the number of points z_j that satisfies $|x_{s+j-1} - x_{s+i-1}| < dx$, which is denoted as $n_x(z_i)$. $n_y(z_i)$ can be calculated similarly.

We illustrate it with the example in Figure 2. We want to compute the MI value of the time window $< 8, 10, 0 >$. First, KSG transforms them into 10 two-dimensional data points in Figure 2(b), $z_i = (x_{7+i}, y_{7+i}), 1 \le i \le 10$. We illustrate the process of computing $n_x(z_1)$ and $n_y(z_1)$. Assume $k = 2$. z_3 and z_7 are 2 nearest neighbours of z_1. So based on them, we can obtain dx and dy. Points z_8, z_9 and z_{10} are located within dx in the X direction, so $n_x(z_1) = 5$. Similarly, we can obtain $n_y(z_1) = 2$ since only two point, z_3 and z_7 are located within the dy region. Next we do the same operations for the rest points, z_i $(2 \le i \le 10)$, and finally get $MI = 0.3473$.



(a) Window pair < 8, 10, 0 >



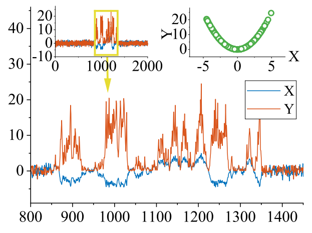
(b) MI Calculation

Figure 2: MI calculation with KSG estimator

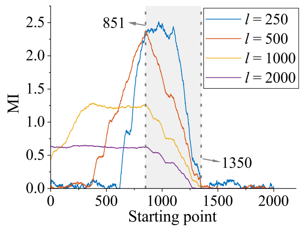
Table 1: Runtime comparison

PCC(brute-force)	MI(brute-force)	Jocor	NLC$_E$	NLC$_S$
812.89	21120.23	2.45	7.57	0.95

$|X| = |Y| = 1000, l \in [300, 500], \tau \in [-100, 100]$



(a) Example time series pair



(b) Influence of s and l

Figure 3: Illustrative example of $\tau = 0$

2.2 Mutual Information vs. Pearson Coefficient

The problem complexity depends on the size of the search space and the cost of MI calculation. Let n be the length of the time series, τ and l be the average delay and length of the window pair respectively. We need to search $O(\tau nl)$ window pairs in total, i.e., the size of the search space. When computing MI for a length-l window pair with KSG estimator, the basic kNN algorithm requires $O(l^2)$, which can be reduced to $O(l \log l)$ by utilizing the k-d tree structure. Therefore, the overall time complexity of the problem is $O(\tau nl^2 \log l)$. If we measure the correlation with PCC, the search space is the same and the unit cost becomes l. So, the overall complexity is $O(\tau nl^2)$.

Although the difference is only $\log l$, it actually takes dozens of times longer. In Table 1, we list the runtime of computing PCC and MI with both brute-force approach and the advanced approach. It can be seen that MI search is magnitude of order slower than PCC search. The core reason is that MI computation needs to find the k nearest neighbours of each data point. Jocor [18] accelerates PCC pair searching by frequency domain transformation and incremental computation. However, the speedup techniques of Jocor cannot be applied to MI computation.

3 MOTIVATION

As mentioned beforehand, the core challenge is the huge time cost to verify all window pairs. Since each time window pair is determined by three factors, starting point, window length and time delay, we analyze the influence of these factors on the MI value.

3.1 Starting Point and Window Length

We start with a simple case with only one aligned correlated window pair in (X, Y), as shown in Figure 3(a). The length of X and Y is 2000. In window $W = [851, 1350]$, X_W and Y_W satisfy $y = x^2$, while in the remaining parts, X and Y conform to the Gaussian distributed ($X \sim N(0, 1)$ and $Y \sim N(0, 1)$). We try to find correlated window W [1]. We show 4 curves in Figure 3(b), each of which is the MI value sequence of the sliding windows with 4 different window lengths, i.e., $l = 250, 500, 1000, 2000$. Given any starting point s and length l, the corresponding window is $W' = (s, l)$. We enumerate the relationship between W' and the target window W in Table 2.

Table 2: Correlation type

Case	l	s	rel with W	MI
1.1		$s < 600 \vee s > 1350$	$W' \cap W = \emptyset$	0
1.2	250	$s \in [851, 1100]$	$W' \subset W$	2.4
1.3		$s \in [600, 850] \vee [1100, 1350]$	$W' \cap W \neq \emptyset$	[0,2.4]
2.1		$s < 350 \vee s > 1350$	$W' \cap W = \emptyset$	0
2.2	500	$s = 851$	$W' = W$	2.3
2.3		$s \in [350, 850]$	$W' \cap W \neq \emptyset$	[0,2.3]
3.1		$s > 1350$	$W' \cap W = \emptyset$	0
3.2	1000	$s \in [351, 851]$	$W' \supset W$	1.25
3.3		$s \in [1, 350] \vee [852, 1350]$	$W' \cap W \neq \emptyset$	[0,1.25]
4.1		$s > 1350$	$W' \cap W = \emptyset$	0
4.2	2000	$s \in [1, 851]$	$W' \supset W$	0.64
4.3		$s \in [852, 1350]$	$W' \cap W \neq \emptyset$	[0,0.64]

According to Figure 3(b), the relationship between W' and W, as well as the MI values computed with W', can be categorized into 4 types as follows,

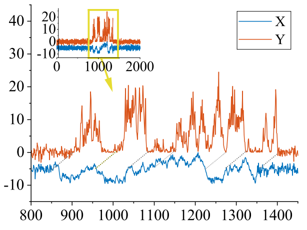
- Case 1.1, 2.1, 3.1 and 4.1. MI values are close to 0, since $X_{W'}$ and $Y_{W'}$ are independent.
- Case 1.2 and 2.2. In these cases, W' is surrounded by W. The MI value reaches the maximum because in the whole window W', X and Y are always correlated.
- Case 3.2 and 4.2. In contrast, when W is surrounded by W', The MI values reach the maximum only within their own lengths (1.25 for $l = 1000$ and 0.64 for $l = 2000$). Note that these two values are smaller than those of case 1.2 and 2.2.
- Case 1.3, 2.3, 3.3 and 4.3. For each length l, the MI values are smaller than the maximum of this length, because W' only contains the partial of W. Moreover, the larger portion of W is contained in W', the larger the MI value.

The above observations give us a clue motivating our approach. We can use the window larger than L_{max} to prune the unnecessary windows. Specifically, assume $L_{max} = 800$ and W_0 is a length-2000 window. If we find that $I(X_{W_0}, Y_{W_0})$ is close to 0, we can determine that there doesn't exist any qualified window W_1 within W_0 ($|W_1| < |W_0|$). Therefore, we can prune plenty of windows.

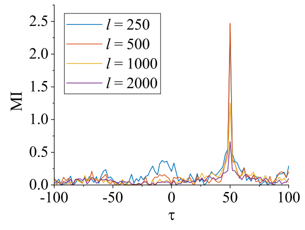
[1]Since the correlated window pair, X_W and Y_W, is aligned, we briefly call to find the correlated window W.

3.2 Delay

Now we come to a more complicated case where the correlated windows are not aligned, that is, $\tau \neq 0$. Continue the example in Figure 3. We fix X unchanged and shift Y to left by 50 points, as shown in Figure 4(a). That is, $X_{[851,1350]}$ is correlated with $Y_{[801,1300]}$.



(a) Example time series pair



(b) Influence of delay τ

Figure 4: Illustrative example of $\tau \neq 0$

Figure 4(b) shows the largest MI values as the delay varies for different window lengths. For example, in the curve of $l = 250$, for each delay value τ, we compute all MI values between $X(i, 250)$ and $Y(i + \tau, 250)$ $(1 \leq i \leq 1751)$ and the maximal one is shown in Y-axis. It can be seen that all 4 curves have the common trends, that is, the values are almost 0 before $\tau = 50$, burst sharply at this point, and then decrease to 0 quickly.

This example reveals a property of the MI computation. If two equal-length windows with delay τ, W_1 and W_2, are correlated. Then two larger window with delay τ, which surround W_1 and W_2 respectively, will be also correlated.

Combining with the previous observation, we give the rationale behind our window shrinking strategy. To search for correlated pairs with appropriate starting points and delays, we use the large window to prune the unqualified window pairs. Moreover, to find the optimal delay value, we directly compute MI values with different delay values on the large window. For a fixed large window W_0, if all possible value of τ cannot generate a correlated window pair, we can prune all window pairs surrounded by W_0.

To be specific, for window $W_0 = (s, 2000)$, we compute $I(X_{W_0}, Y(s+j, 2000))$ for any $j \in [\tau_{min}, \tau_{max}]$. If all MI values are small, we can prune all windows surrounded by W_0. In contrast, if there exists $\tau \in [\tau_{min}, \tau_{max}]$, which satisfies that $I(X_{W_0}, Y_{(S+\tau,2000)})$ is obviously large, it means that there may exist two windows, $W_1 = (s', l)$ and $W_2 = (s' + \tau, l)$, which are correlated.

In other words, we generate candidates with fuzzy starting points and lengths, but *exact* time delay. Another reason to support our choice is that the MI values fluctuate more quickly as time delay varies, compared to starting point and length.

This strategy works well when the correlated pairs disperse sparsely. However, when they distribute more densely, this strategy may suffer because the large window is likely to contain more than one correlated pairs. The consequence is that we may only obtain the dominant pair in a large window, and ignore other pairs.

To avoid this problem, we propose another strategy, window extending, to generate candidates for the densely distributed pairs. Note that in examples of both Figure 3 and Figure 4, windows W' of length 250, although it is smaller than 500, can still achieve high MI values. So, in the extending strategy, we use length-L_{min} windows to make pruning. Specifically, for window $W_0 = (s, L_{min})$, we compute $I(X_{W_0}, Y(s+j, L_{min}))$ for any $j \in [\tau_{min}, \tau_{max}]$. If all MI values are small, we can prune all windows surrounding W_0.

3.3 Theoretical Foundation

In this section, we present the theoretical foundation of the proposed approach. We formally analyze the relationship between MI value of the correlated pair and a larger pair surrounding it.

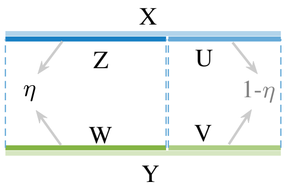


Figure 5: Window pair (X, Y) and (Z, W)

THEOREM 1. *Given a window pair (X, Y), X is spliced by Z and U, and Y is spliced by W and V. The length ratio between $|Z|$ and $|X|$ (or $|W|$ and $|Y|$) is η. Assume Z and U are identically distributed, and the value ranges of W and V are disjoint, $R_W \cap R_V = \emptyset$, then it holds that*

$$I(X;Y) = \eta I(Z;W) + (1-\eta) I(U;V)$$

As a special case, when U and V are independent, it holds that

$$I(X;Y) = \eta I(Z;W)$$

Proof. From a statistical perspective, the probability density functions are $p_Z(z), p_W(w), p_U(u), p_V(v)$ for Z, W, U, V and $p_{Z,W}(z,w)$, $p_{U,V}(u,v)$ for (Z,W), (U,V). Since Z and U are identically distributed, the probability of X satisfies $p_X(x) = p_Z(x) = p_U(x)$.

Because Y is spliced by W and V, the probability of Y is

$$p_Y(y) = \eta p_W(y) + (1-\eta) p_V(y) \tag{3}$$

The joint probability $p_{X,Y}(x,y)$ can be written as:

$$p_{X,Y}(x,y) = \eta p_{Z,W}(x,y) + (1-\eta) p_{U,V}(x,y) \tag{4}$$

We define the value range of W and V as R_W and R_V, the value range of (X,Y), (Z,W) and (U,V) as R_{XY}, R_{ZW} and R_{UV} respectively. Considering that $R_{XY} = R_{ZW} \cup R_{UV}$, the *MI* between X and Y is as follows:

$$\begin{aligned} I(X;Y) &= \sum_{(x,y)\in R_{XY}} p_{X,Y}(x,y) log \frac{p_{X,Y}(x,y)}{p_X(x)p_Y(y)} \\ &= \sum_{(z,w)\in R_{ZW}} p_{X,Y}(z,w) log \frac{p_{X,Y}(z,w)}{p_X(z)p_Y(w)} \\ &+ \sum_{(u,v)\in R_{UV}} p_{X,Y}(u,v) log \frac{p_{X,Y}(u,v)}{p_X(u)p_Y(v)} \\ &- \sum_{(x,y)\in R_{ZW}\cap R_{UV}} p_{X,Y}(x,y) log \frac{p_{X,Y}(x,y)}{p_X(x)p_Y(y)} \end{aligned} \tag{5}$$

We substitute the joint probability $p_{X,Y}(x,y)$ with Eq. 4 and get

$$\begin{aligned} I(X;Y) = &\sum_{(z,w)\in R_{ZW}} [\eta p_{Z,W}(z,w) + (1-\eta) p_{U,V}(z,w)] log \frac{p_{X,Y}(z,w)}{p_X(z)p_Y(w)} \\ + &\sum_{(u,v)\in R_{UV}} [(1-\eta) p_{U,V}(u,v) + \eta p_{Z,W}(u,v)] log \frac{p_{X,Y}(u,v)}{p_X(u)p_Y(v)} \\ - &\sum_{(x,y)\in R_{ZW}\cap R_{UV}} p_{X,Y}(x,y) log \frac{p_{X,Y}(x,y)}{p_X(x)p_Y(y)} \end{aligned} \tag{6}$$

Then we expand the formula to get

$$\begin{aligned} I(X;Y) = &\sum_{(z,w)\in R_{ZW}} \eta p_{Z,W}(z,w) log \frac{p_{X,Y}(z,w)}{p_X(z)p_Y(w)} \\ + &\sum_{(u,v)\in R_{UV}} (1-\eta) p_{U,V}(u,v) log \frac{p_{X,Y}(u,v)}{p_X(u)p_Y(v)} \\ + &\sum_{(z,w)\in R_{ZW}\cap R_{UV}} (1-\eta) p_{U,V}(z,w) \frac{p_{X,Y}(z,w)}{p_X(z)p_Y(w)} \\ + &\sum_{(u,v)\in R_{UV}\cap R_{ZW}} \eta p_{Z,W}(u,v) log \frac{p_{X,Y}(u,v)}{p_X(u)p_Y(v)} \\ - &\sum_{(x,y)\in R_{ZW}\cap R_{UV}} p_{X,Y}(x,y) log \frac{p_{X,Y}(x,y)}{p_X(x)p_Y(y)} \end{aligned} \tag{7}$$

The third term in Eq. 7 is derived from the first term in Eq. 6. Based on $(x,y) \in R_{XY}$, here $p_{U,V}(x,y)$ implies that the (x,y) is actually from $R_{XY} \cap R_{UV}$. The fourth term in Eq. 7 is the same. According to Eq. 4, we have the sum of the last 3 terms of Eq. 7 is zero.

Since $p_X(x) = p_Z(x) = p_U(x)$, by simple algebraic transformation, we have

$$\begin{aligned} I(X;Y) = &\sum_{(z,w)\in R_{ZW}} \eta p_{Z,W}(z,w) log[\frac{p_{Z,W}(z,w)}{p_Z(z)p_W(w)} \cdot \frac{p_W(w)}{p_Y(w)} \cdot \frac{p_{X,Y}(z,w)}{p_{Z,W}(z,w)}] \\ + &\sum_{(u,v)\in R_{UV}} (1-\eta) p_{U,V}(u,v) log[\frac{p_{U,V}(u,v)}{p_U(u)p_V(v)} \cdot \frac{p_V(v)}{p_Y(v)} \cdot \frac{p_{X,Y}(u,v)}{p_{U,V}(u,v)}] \end{aligned} \tag{8}$$

We expand the formula:

$$\begin{aligned} I(X;Y) = &\sum_{(z,w)\in R_{ZW}} \eta p_{Z,W}(z,w) log \frac{p_{Z,W}(z,w)}{p_Z(z)p_W(w)} \\ + &\sum_{(z,w)\in R_{ZW}} \eta p_{Z,W}(z,w) log[\frac{p_W(w)}{p_Y(w)} \cdot \frac{p_{X,Y}(z,w)}{p_{Z,W}(z,w)}] \\ + &\sum_{(u,v)\in R_{UV}} (1-\eta) p_{U,V}(u,v) log \frac{p_{U,V}(u,v)}{p_U(u)p_V(v)} \\ + &\sum_{(u,v)\in R_{UV}} (1-\eta) p_{U,V}(u,v) log[\frac{p_V(v)}{p_Y(v)} \cdot \frac{p_{X,Y}(u,v)}{p_{U,V}(u,v)}] \end{aligned} \tag{9}$$

We substitute $I(Z;W)$ and $I(U;V)$ into the first and third terms of Eq. 9, and finally get:

$$\begin{aligned} I(X;Y) = &\eta I(Z;W) + (1-\eta) I(U;V) \\ + &\sum_{(z,w)\in R_{ZW}} \eta p_{Z,W}(z,w) log[\frac{p_W(w)}{p_Y(w)} \cdot \frac{p_{X,Y}(z,w)}{p_{Z,W}(z,w)}] \end{aligned}$$

$$+ \sum_{(u,v)\in R_{UV}} (1-\eta)p_{U,V}(u,v)log[\frac{p_V(v)}{p_Y(v)} \cdot \frac{p_{X,Y}(u,v)}{p_{U,V}(u,v)}] \quad (10)$$

Since we assume $R_W \cap R_V = \emptyset$, any value of Y is taken from either R_W or R_V, i.e. Eq. 3 changes to $p_Y(w) = \eta p_W(w)$ and $p_Y(v) = (1-\eta)p_V(v)$. Similarly, any value pair of (X, Y) is taken from either R_{ZW} or R_{UV}. In consequence, we have $p_{X,Y}(z,w) = \eta p_{Z,W}(z,w)$ and $p_{X,Y}(u,v) = (1-\eta)p_{U,V}(u,v)$ respectively.

By combining these 4 formulas with Eq. 10, we have

$$I(X;Y) = \eta I(Z;W) + (1-\eta)I(U;V) \quad (11)$$

If U and V are independent, then $I(U;V) = 0$ and $I(X;Y) = \eta I(Z;W)$.

In the example of Figure 3, we can consider range $[851, 1350]$ as (Z, W) here and other parts as (U, V). Since values of Y in $[851, 1350]$ and those in other parts are almost non-overlapping, the MI relationships in Table 2 are consistent with Theorem 1.

3.4 Overview

We propose a two-phase approach to search the correlated pairs, as shown in Figure 6. In phase one, we scan X and Y sequentially and generate a candidate set , denoted as $\mathcal{CP}_0$. In phase two, we refine the correlated pairs in $\mathcal{CP}_0$ to form the final set $\mathcal{CP}$. Each window pair in $\mathcal{CP}$, $CP =< s, l, \tau >$, is derived from a pair in $\mathcal{CP}_0$, $CP' =< s', l', \tau' >$, which satisfies that they have the same delay, that is, $\tau = \tau'$. These two phases are outlined as follows.

- **Phase one: Candidate generation.** In this phase, we generate the candidate set $\mathcal{CP}_0$. Each candidates determine 1) the approximate location and window length, 2) the *exact* delay. We propose two strategies to generate $\mathcal{CP}_0$, window shrinking and window extending.
- **Phase two: Window refinement.** We refine candidates in $\mathcal{CP}_0$ to obtain the final results $\mathcal{CP}$. We consider it as an optimization problem, and adapt the DIRECT strategy [14] to propose a nested one-dimensional search algorithm.

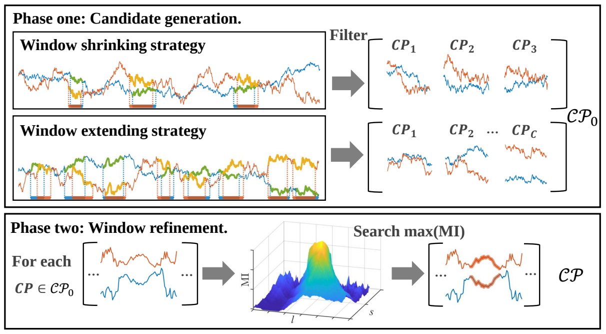


Figure 6: Algorithm overview

4 PHASE ONE: CANDIDATE GENERATION

To generate the candidate set $\mathcal{CP}_0$, we propose two strategies, window shrinking and window extending. The former first finds the large window pairs which may contain the qualified correlated pair, and shrinks the windows by trimming the uncorrelated parts. In contrast, the latter first finds the small window pairs, and extends them to appropriate lengths. Next, we introduce them in turn.

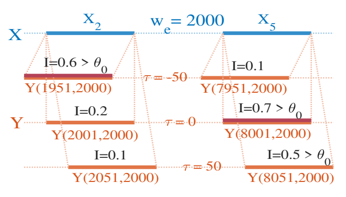
4.1 Window Shrinking

The window shrinking strategy is used to deal with the sparsely distributed correlations, whose pseudo-code is shown in Algorithm 1.

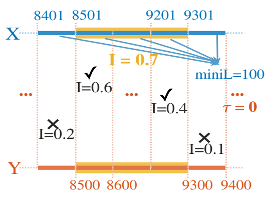
Note that our goal is to find all correlated window pairs whose size fall within $[L_{min}, L_{max}]$. Instead of computing *MI* of each pair directly, we utilize a window longer than L_{max} to filter unqualified candidates, named as envelope window, whose size is denoted as w_e. Formally, we split X into a sequence of disjoint envelope windows with length w_e, $(X_1, X_2, \cdots, X_{\frac{n}{w_e}})$, where $X_i = X(sp_i, w_e)$ and $sp_i = (i-1) * w_e + 1$ (Line 2). We assume in each window X_i, there exists at most one smaller window correlated with Y.

Algorithm 1 Window shrinking

```
Input: $(X, Y)$: a pair of time series
Output: $\mathcal{CP}_0$: the candidate set
1: while $(X, Y)$ is not scanned entirely do
2:     Initialize $CP \longleftarrow < sp_i, w_e, 0 >$ with $I_{best} = 0$
3:     for $\tau \in [\tau_{min}, \tau_{max}]$ do
4:         $EW \longleftarrow (X_i, Y(sp_i + \tau, w_e))$
5:         if $I(EW) > I_{best}$ then
6:             $I_{best} \longleftarrow I(EW)$, update $CP \longleftarrow < sp_i, w_e, \tau >$
7:     if $I_{best} > \theta_0$ then
8:         $CP.trim(miniL, \theta_0)$
9:         add $CP$ to $\mathcal{CP}_0$
```



(a) Envelope window filter process



(b) The trim operation

Figure 7: Window shrinking

Then we visit the envelope windows one-by-one to determine whether it may contain a candidate pair (Line 3-6). For each X_i, we compute MI values between X_i and all windows $Y(sp_i + \tau, w_e)$'s ($\tau_{min} \le \tau \le \tau_{max}$). If all MI values are smaller than threshold θ_0, we determine that there is no correlated pair here and move to X_{i+1}. Otherwise, we select the window $Y(sp_i + \tau', w_e)$ with the largest MI value as a candidate $CP =< sp_i, w_e, \tau' >$. It means that it is very likely that there exists one window within X_i, which is correlated with a window in $Y(sp_i + \tau', w_e)$ with delay τ'.

Figure 3 shows that when the current envelope window W_e contains the correlated window pair W_c (other parts are completely uncorrelated), the correlation length ratio is approximately the same as the MI value, *i.e.*, $I(W_e)/I(W_c) \approx |W_e|/|W_c|$. Since the correlation length is at least L_{min}, the envelope window length is w_e, θ_0 is set to $\frac{\theta L_{min}}{w_e}$.

We illustrate it with the example in Figure 7(a), in which θ_0 is set to 0.3 and $w_e = 2000$. In window $X_2 = (2001, 2000)$, $Y(2001, 2000)$

and $Y(2051, 2000)$ will be pruned since the MI values are smaller than 0.3. Since $Y(1951, 2000)$ has the highest MI value, we generate a candidate $< 2001, 2000, -50 >$. Similarly, in window X_5, both $Y(8001, 2000)$ and $Y(8051, 2000)$ have MI values higher than 0.3, and finally the candidate is $< 8001, 2000, 0 >$.

Obviously, candidate $CP =< s, l, \tau >$ may contain uncorrelated parts, since it has a large window length and a smaller threshold θ_0. So we need to *trim* these parts (Line 8). To be specific, we split the envelop window into a sequence of smaller disjoint windows, and trim the uncorrelated windows on both ends. Formally, we first divide the envelop window into windows of length $miniL$, named as *mini window*. Then, we trim the unqualified mini windows from the left end. We calculate the MI value between $X(s, miniL)$ and $Y(s + \tau, miniL)$. If the MI value is less than θ_0, we discard it and move to the second mini window. This process continues until we meet a window with MI value larger than θ_0. Then we trim mini windows from the right end with a similar manner. After trimming on both ends, we again calculate the MI value of the remaining parts, and add it into $\mathcal{CP}_0$ if the MI value exceeds θ_0.

Continue the example in Figure 7(b). The envelop window is $< 8001, 2000, 0 >$ and $miniL = 100$. On the left, we trim mini windows until $(8501, 100)$, while on the right, we trim windows until $(9301, 100)$. The rest part is $(8501, 800)$. We recalculate MI and get $I = 0.7 > \theta_0$, so we add $< 8501, 800, 0 >$ to $\mathcal{CP}_0$.

4.2 Window Extending

Now we introduce our second strategy, window extending. When the correlated pairs are densely distributed, if we still use the large window, it may occur that two different correlated pairs appear in one envelop window, which will cause certain correlated pair to be ignored since we only keep one pair in an envelop window.

So the window extending strategy is to use small window to find the candidates. The pseudo-code is shown in Algorithm 2. Formly, we use the length-L_{min} windows to traverse X (Line 2). Assume the current window is $X(sp, L_{min})$. We compute MI values of all windows $Y(sp + \tau, L_{min})$'s, where $\tau_{min} \leq \tau \leq \tau_{max}$. If all MI values are smaller than threshold θ_0, we determine that there is no correlated pair here and move to the next window $X(sp + L_{min}, L_{min})$ (Line 14). Otherwise, we select τ which leads to the maximal MI value to form a pair $< sp, L_{min}, \tau >$. Here θ_0 is set to $\frac{\theta L_{min}}{L_{max}+L_{min}}$, with the same reason as shrinking strategy.

Different from the the window shrinking, the correlated pairs we find here may be only a fraction of the whole pair due to the small window size. So, we try to find the complete pair by extending the window on both ends. Formally, for window pair $X(sp, L_{min})$ and $Y(sp + \tau, L_{min})$, we generate two candidates,

$$CP_1 =< sp, L_{min} + L_{max}, \tau >$$

$$CP_2 =< sp - L_{max}, L_{min} + L_{max}, \tau >$$

Then we do the trim operation on both CP_1 and CP_2, and obtain one shorter pairs (Line 7,8). We compare their corresponding MI values and add the one with larger MI into $\mathcal{CP}_0$ (Line 10,12).

We illustrate the window extending strategy with Figure 8, where $\theta_0 = 0.3$. The length constraint is $[300, 600]$, so we use the length-300 windows for pruning. The current window $X(8501, 300)$ achieves the highest MI value when $\tau = 0$. Then we extend the pair to get two candidates, $CP_1 =< 8501, 900, 0 >$ and $CP_2 =< 7901, 900, 0 >$. After trimming, CP_2 gets higher MI, so we add it to $\mathcal{CP}_0$.

Algorithm 2 Window extending

Input: (X,Y): a pair of time series
Output: R^L: a set of located window pairs

```
1:  while $(X, Y)$ is not scanned entirely do
2:      Initialize $CP \longleftarrow < sp, L_{min}, 0 >$ with $I_{best} = 0$
3:  ◀ Follow the steps 3-6 in Algorithm 1 to get $I_{best}$ and update $CP$
4:      if $I_{best} \geq \theta_0$ then
5:          $CP_1 \longleftarrow < sp, L_{min} + L_{max}, \tau >$,
6:          $CP_2 \longleftarrow < sp - L_{max}, L_{min} + L_{max}, \tau >$
7:          $CP_1.trim(miniL, \theta_0)$
8:          $CP_2.trim(miniL, \theta_0)$
9:          if $I(CP_1) \geq I(CP_2)$ then
10:             Add $CP_1$ to $\mathcal{CP}_0$, update $sp \leftarrow sp + |CP_1|$
11:         else
12:             Add $CP_2$ to $\mathcal{CP}_0$, update $sp \leftarrow sp + |CP_2|$
13:     else
14:         $sp \leftarrow sp + L_{min}$
```

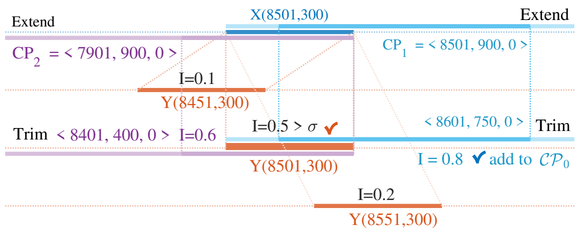


Figure 8: Window extending

5 PHASE TWO: WINDOW REFINEMENT

In phase two, we refine the candidates in $\mathcal{CP}_0$, by taking it as an optimization problem. Formally, for each candidate pair $CP =< s, l, \tau >$, we aim to find $< s^*, l^*, \tau >$, which satisfies, 1) window $(s^*, l^*) \subset (s, l)$, and 2) $I(s^*, l^*, \tau)$ is the largest. We call s^* and l^* as the optimal starting point and window length respectively. The naive approach is to enumerate all possible s and l and select the optimal one. However, it is infeasible due to the high computation cost. In this paper, we adapt the popular DIRECT strategy [14] to accelerate the searching process.

In general, DIRECT is a sampling-based search strategy for the optimization problem. For an objective function over a search space, DIRECT can effectively find the global optimum through an iterative process. DIRECT is kind of a ternary search strategy. The basic idea is to partition the search space into intervals, and select the rectangles to search next by sampling and evaluating the center points of each interval.

For candidate pair $CP =< s, l, \tau >$, the search space is a rectangle: one dimension for the starting point s and the other for length l. To ensure that the window length is always legal, the range of the starting point is $[s, s + l - L_{min}]$, and the range of l is $[L_{min}, L_{max}]$. For multivariate variables, DIRECT divide dimensions in turn to ensure the balance of sampling in each dimension. However, the fashion of one-dimension-per-time is not suitable for our problem because the starting point s and length l are highly correlated.

To solve this problem, we propose a nested one-dimensional DIRECT search strategy, that is, we try to find the optimal s^*. The basic idea is as follows. In each round, we split the current rectangle into three equal-size rectangles, left, middle and right. Then we shrink the rectangle by removing the left or right one, which is less likely to contain point (s^*, l^*). The decision is made by evaluating the MI values of the center points s. Note that the range of length l is kept unchanged during the iteration.

Our approach works as follows. Initially, the range of s is $[lb, ub] = [s, s + l - L_{min}]$. In the first round, we split $[lb, ub]$ into three equal length ranges, $S_1 = [lb, lb + \frac{ub-lb}{3}]$, $S_2 = [lb + \frac{ub-lb}{3}, ub - \frac{ub-lb}{3}]$ and $S_3 = [ub - \frac{ub-lb}{3}, ub]$. We compute the optimal MI values of the central values of s, s_1 and s_3, in S_1 and S_3. That is, for each s, we find the optimal length $l^*(s)$ satisfying

$$l^*(s) = \max_{l \in [L_{min}, L_{max}]} I(s, l, \tau)$$

How to find $l^*(s)$ will be discussed later. After we obtain $I(s_1, l^*(s_1), \tau)$ and $I(s_3, l^*(s_3), \tau)$, we compare them and remove S_1 (or S_3) from $[lb, ub]$ if the former is lower (or larger). Then we use the new range to begin the second round. This process until lb and ub is smaller than a threshold, denoted as δ_s.

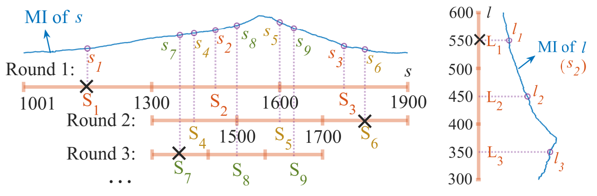


Figure 9: Search process

The process of finding $l^*(s)$ for each s is similar to finding s^*, except that we can directly compute the MI values for the sampled lengths. The stopping threshold is δ_l. After we find the optimal s^*, we finally verify whether it is a qualified pair, that is, $I(s^*, l^*, \tau) \geq \theta$. If it is the case, we add it into the final set $\mathcal{CP}$. Note that the candidates generated by the extending strategy may overlap. We adjust candidates according to the qualified CPs that have been refined to ensure that there is no overlap in $\mathcal{CP}$.

We use an example in Figure 9 to illustrate the process. In the left part, the blue curve is the MI values of different starting points in range $[1001, 1900]$. We aim to find the optimal starting points s^* with highest MI value within this range. In the first round, we split it into three equal-length ranges $S_1 = [1001, 1300]$, $S_2 = [1301, 1600]$, $S_3 = [1601, 1900]$, and obtain three center points, s_1, s_2 and s_3. Then we find l^* for s_1 and s_3, which will be described later. After we found them, we compare MI values $I(s_1, l^*(s_1))$ and $I(s_3, l^*(s_3))$. Since MI value of s_1 is less than that of s_3, we remove S_1 and generate a smaller range $[1301, 1900]$. In the second round, we again split range $[1301, 1900]$ into three ranges, S_4, S_5 and S_6, with center points s_4, s_5 and s_6. It can be seen that after three rounds, the MI value at s_5 is very close to the highest.

In the right part of Figure 9, we show the range splitting in the first round of searching $l^*(s_2)$. We split range $[300, 600]$, which are L_{min} and L_{max}, into three ranges L_1, L_2 and L_3 with center points $l_1 = 550$, $l_2 = 450$ and $l_3 = 350$. We directly compute $I(s_2, 550)$ and $I(s_2, 350)$, and then remove L_1 since $I(s_2, 550)$ is smaller. Then, we iteratively search range $[300, 500]$ to find $l^*(s_2)$.

Complexity analysis. Now we analyze the complexity of the algorithm. In phase one, we filter the candidate with two strategies. The shrinking strategy split (X, Y) into length-w_e envelope windows. Here the cost is $O(\frac{n}{w_e}\tau l \log l)$. The extending strategy uses length-L_{min} windows, so the cost is $O(\frac{n}{L_{min}}\tau l \log l)$.

In phase two, we use a nested ternary search to refine the candidates. Since the size of the search rectangle is approximately l^2, we need $O(l(\log l)^3)$. If there are m candidates in $\mathcal{CP}$, the total cost is $O(ml(\log l)^3)$. Overall, our algorithm reduces the problem complexity from $O(n\tau l^2 \log l)$ to $O(\frac{n}{w_e}\tau l \log l + ml(\log l)^3)$ or $O(\frac{n}{L_{min}}\tau l \log l + ml(\log l)^3)$.

6 EXPERIMENTAL RESULTS

In this section, we conduct extensive experiments to verify the efficiency and effectiveness of the proposed approach. All experiments are conducted on are on a standard PC with 3.0 GHz processor, 16 GB RAM and 256 GB SSD.

6.1 Experimental Setting

6.1.1 *Dataset.*

We use five datasets for experiments, including two synthetic datasets and three real-world datasets.

Synthetic Datasets. Two synthetic datasets are generated with linear and non-linear correlations respectively. Initially, we generate two independent time series X and Y of length 100000, where the values are randomly picked from a standard normal distribution. Then we plant correlations as follows.

Non-linear dataset. Table 3 shows ten types of planted correlations, logarithmic, exponential, square and so on. To plant a correlated pair, we execute the following two steps. (i) We randomly select a starting point, and a non-linear relation type t from Table 3. (ii) Then a correlated window pair CP $(l \in [400, 850] \wedge \tau \in [-150, 150])$ is generated and inserted into (X, Y) according to t. We plant 40 correlated window pairs.

Linear dataset. The data generation process is similar to the non-linear dataset with the same length constraint, delay range and data scale, but the correlation type is always Linear (Table 3).

Table 3: Correlations in different relations

Relations	$y = f(x)$
Independent	$x \sim N(0,1), y \sim N(0,1)$
Linear	$y = 2x + u, x \sim U(-5,5)$
Log.	$y = ln(tx), x \sim U(0,10)$
Exp.	$y = 0.1^x, x \sim U(-10,10)$
Square	$y = x^2 + u, x \sim U(-4,4)$
Squareroot	$y = sqrt(x + u * x), x \sim U(0,25), u \sim U(5,10)$
Circle	$y = sqrt(9 - x^2 + u), x \sim U(-3,3)$
Sine.	$y = 2sin(x) + u, x \sim U(0,10)$
Cross	$y = x + u, x \sim U(-5,5)$
Quartic	$y = x^4 - 4x^3 + 4x^2 + x + u, x \sim U(-1,3)$
Reciprocal	$y = 10/(x + u * x), x \sim U(0,5)$
Trig.	$y = cos(x) + sin(x) + tan(x), x \sim U(-5,5)$

Note: $N(\mu, \sigma)$: normal distribution, $U(a, b)$: uniform distribution, default noise $u \sim U(0, 1)$

Real-world Datasets. Three real-world datasets are collected from different domains, covering industry, economy, and technology.

(1) **TE datast.** Tennessee eastman (TE) process [6] is a realistic simulation model of an industrial chemical process. TE dataset includes 21 abnormal conditions and 1 normal condition, which are connected after z-normalization. There are 52 time series in total, and the length of each is 10580. For pairwise combinations, we actually need to analyze 1326 pairs of time series, with a total length of 14029080.
(2) **Stock market dataset.** Stock market dataset [19] contains historical daily prices for tickers trading on NASDAQ. Here we select 50 well-known company stocks for analysis. Specifically, we select 1500-day closing data during the period from 04/2014 to 04/2020. Most of the selected stocks are in the technology sector, some catering and cultural stocks are also included. Similarly, we make correlation analysis in pairs.
(3) **Electrical dataset.** NIST Net-Zero dataset [8] captures diverse measurements from a high precision lab home. The dataset enables engineers to better understand how to build more energy-efficient homes. We extract electrical system data from 02/2015 to 01/2016. Each time series represents the electricity consumption of a specific object, and each value records the instantaneous electricity consumption in minutes. There are 72 time series, all of which are 518792 in length. We pick 141 time series pairs for correlation analysis.

Table 4: Experimental parameter settings

Parameters / Datasets	$[L_{min}, L_{max}]$	$[\tau_{min}, \tau_{max}]$
(Non-)Linear dataset	[400,850]	[-150,150]
TE dataset	[120,240](mins)	[-60,60](mins)
Stock market dataset	[200,400](days)	[-30,30](days)
Electrical dataset	[40,120](mins)	[-30,30](mins)

6.1.2 Counterpart Approaches.
In this paper, NLC is compared with three baseline methods: TYCOS [9], MASS [22] and Jocor [18].

TYCOS [9] extracts non-linear correlated window pairs from long time sequence pairs, and also uses MI as the correlation coefficient. TYCOS combines Late Acceptance Hill Climbing [3] with window methods to perform bottom-up correlated window pair search. We have reproduced it according to the paper description.

Jocor [18] finds the most correlated pairs with length longer than the minimum length threshold, and uses PCC as the metric. We use the code provided by the author. Since Jocor requires $O(n^2 logn)$ when preprocessing the cross product matrix, which is too slow for long time series. So when $n > 10000$, we divide the entire time series into non-overlapping segments and perform Jocor in each segment, just like the window shrinking strategy.

MASS [22] can efficiently find the subsequence most similar to the query in a long time series. Since the original intention of MASS is different from our problem, some adjustments are needed. Considering the lag constraint $[\tau_{min}, \tau_{max}]$, we take the time interval $w_X = (s, l)$ as the query, and $w_Y = (s + \tau_{min}, l + \tau_{max} - \tau_{min})$ as the long time series. Note that MASS uses the traditional PCC and Euclidean distance as the correlation coefficient, which means that we need to set appropriate linear and non-linear thresholds, which will be discussed in Section 6.5.1.

6.1.3 Implementation and Parameter Settings.
NLC is implemented in MATLAB, and the code is publicly available[2]. The k-d tree structure is used for searching nearest neighbors in MI computation, due to its high efficiency. We use NLC$_E$ and NLC$_S$ to represent NLC with the extending strategy and the shrinking strategy respectively.

All parameters are of two types: problem constraint parameters and algorithm parameters. The former is shown in Table 4. For synthetic datasets, the known window length and delay range of data generation are used as the constraints of our search. For real-world datasets, the parameters are set according to the domain common sense.

The default values of algorithm parameters are as follows. In phase one, we use w_e in the shrinking strategy and $miniL$ in trim operations of both strategies. In phase two, we use δ_l and δ_s as the termination parameter of the search process. $miniL = \frac{L_{min}}{10}$ for both strategies, $w_e = 2L_{max}$ and $\delta_s = \delta_l = \frac{L_{min}}{10}$ for NLC$_E$, $\delta_s = \delta_l = \frac{L_{min}}{2}$ for NLC$_S$. We discuss them in more detail in Section 6.2.

Measurements. The algorithm quality is measured by precision, recall and F1-score. According to the results, all data points in the time series pair are divided into two types: positive and negative. Positive means that the point falls in a correlated window pair, and negative means the opposite. There are four situations for each point. True Positives (TP), means that the true correlated point is marked as positive by the algorithm. False Positives (FP), False negatives (FN) and True negatives (TN) is defined similarly. The quality is evaluated by $Precision = \frac{TP}{TP+FP}$, $Recall = \frac{TP}{TP+FN}$ and $F1 - score = \frac{2*Precision*Recall}{Precision+Recall}$ accordingly.

6.2 Parameter Influences

First, we study the influence of parameters, w_e, $miniL$, δ_s and δ_l. When adjusting one parameter, other parameters keep default values. The used dataset is the synthetic non-linear dataset.

miniL. The trim operation used in phase one is controlled by $miniL$. As shown in Figure 10(a), all F1-score, precision, and recall of NLC$_E$ are always greater than 0.9 when $miniL$ is below $0.5 * L_{min}$. Recall reaches the best when $miniL$ is set to $0.1 * L_{min}$, F1-score and precision are the highest when $miniL = 0.4 * L_{min}$. Without trim operation, the time consumption of NLC$_E$ increases apparently, and the recall and precision drop below 0.9, which also verify the effectiveness of the trim operation.

Figure 10(b) shows that all three indicators of NLC$_S$ are almost unchanged when $miniL$ is less than $0.5 * L_{min}$. Similarly, the efficiency decreases when NLC$_S$ is without trim operation. So we recommend $miniL$ to be set as $0.1 * L_{min}$.

δ_l and δ_s. In phase two, the search termination is determined by δ_s and δ_l. It can be seen from Figure 11 and Figure 12 that as δ_l and δ_s increase, time consumption, recall and F1-score decrease.

[2]https://github.com/zhxjz/NLC

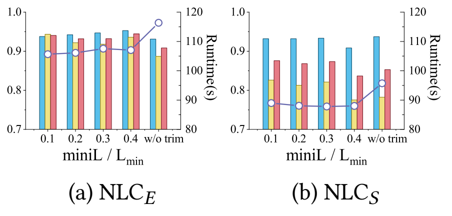


Figure 10: Influence of parameter $miniL$

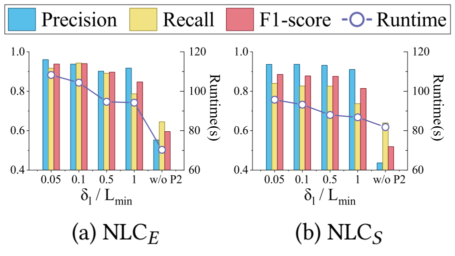


Figure 11: Influence of parameter δ_l

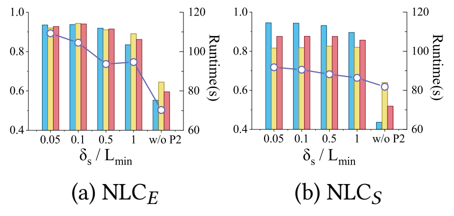


Figure 12: Influence of parameter δ_s

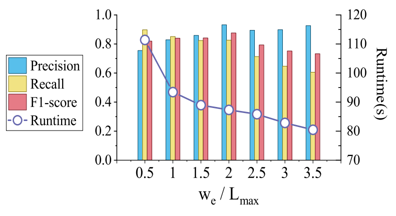


Figure 13: Influence of parameter w_e

The changes caused by δ_l are more significant than δ_s. This is because in our nested one-dimensional search strategy, δ_l are used more frequently. When δ_s and δ_l are set within L_{min}, F1-score is always greater than 0.8. To balance effectiveness and efficiency, the recommended values are $\delta_s = \delta_l = \frac{L_{min}}{10}$ for NLC$_E$, $\delta_s = \delta_l = \frac{L_{min}}{2}$ for NLC$_S$. In fact, both of them perform stable. As long as they are set within a reasonable range, good results will be obtained.

Here, we also compare the performance without phase two. Figure 11 and Figure 12 show that without phase two, the precision drops significantly, which verify its effectiveness.

w_e. Results are shown in Figure 13. It can be seen that w_e is a trade-off between precision and recall. As w_e becomes larger, precision increases and recall decreases gradually. Moreover, the larger the w_e, the less time it takes. To balance efficiency and accuracy, the recommended value of w_e is $2 * L_{max}$.

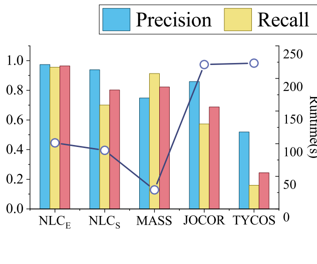


Figure 14: Linear dataset

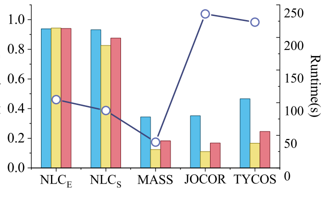


Figure 15: Non-linear dataset

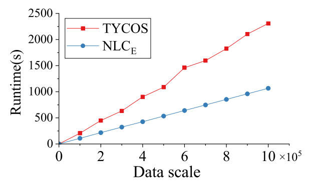


Figure 16: Efficiency

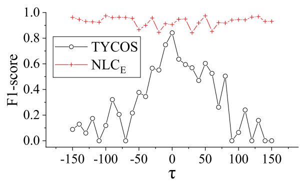


Figure 17: Delay influence

6.3 Results on Synthetic Datasets

In this experiment, we compare NLC with other approaches on synthetic datasets.

6.3.1 *Linear dataset results.*

In this experiment, we compare NLC$_E$ and NLC$_S$ with MASS, Jocor and TYCOS. Since MI and PCC are both used, a unified standard needs to be set. Among all the known correlated pairs, the minimum PCC and MI are used as the threshold for linear and non-linear methods respectively.

The experimental results are shown in Figure 14. Except TYCOS, all other approaches perform well for the linear correlation. Moreover, NLC$_E$ achieves the highest precision, recall and F1-score. Precision of NLC$_S$ is the second highest, but it is more efficient than NLC$_E$.

MASS has the second highest F1-score, but precision is slightly lower. This is because MASS tends to find more windows which may contain noise. Jocor's precision is very high while recall is lower than MASS due to the reduced search space. And the runtime is much larger except TYCOS due to the preprocessing step. TYCOS's poor performance here is mainly because it only tries a few possible values of the delay parameters.

6.3.2 *Non-linear dataset results.*

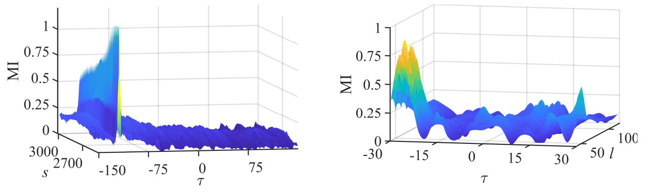
Experimental results are shown in Figure 15. NLC_E achieves the best on all three indicators, and NLC_S is in the second place. Different from the result on the linear dataset, MASS and Jocor, although more efficient, work poor when dealing with non-linear correlations, which demonstrates the advantage of the MI-based approach. Compared to NLC_E, the merit of NLC_S is that it is more efficient, and also can achieve high F1-socre when the density of correlation pairs is low, Table 5 gives the more detailed comparison.

We further compare NLC$_E$ with TYCOS on the non-linear dataset. Figure 16 shows the runtime results as the data size increases. It can be seen that NLC$_E$ is much more efficient than TYCOS. In Figure 17, we compare F1-score for different time delay of the planted correlated pairs. NLC$_E$ always maintains high F1-score no matter the time delay. In contrast, the F1-score of TYCOS decreases quickly as $|\tau|$ increases. This result also verifies our observation that τ has larger influence on the MI value.

We also compare the influence of s, l and τ on the MI value in Figure 18. It can be found that in both synthetic datasets and real-world datasets, the delay τ clearly has a greater influence on MI value, compared with s and l. In other words, if τ is correct but s and l are wrong, it is still very hopeful to find the correlated

Table 5: Comparison between NLC$_E$ and NLC$_S$

Density	NLC$_E$				NLC$_S$ (w_e = 3000)				NLC$_S$ (w_e = 4000)			
	Runtime(s)	Precision(%)	Recall(%)	F1-score	Runtime(s)	Precision(%)	Recall(%)	F1-score	Runtime(s)	Precision(%)	Recall(%)	F1-score
0.05	38.284	98.39	98.78	0.9858	25.849	97.20	98.22	0.9771	24.945	95.52	88.46	0.9186
0.1	41.155	98.14	97.92	0.9803	28.979	97.98	81.02	0.8869	29.051	95.60	76.54	0.8501
0.15	39.538	96.63	97.53	0.9708	32.506	97.39	73.92	0.8405	30.879	96.09	71.35	0.8189
0.2	41.437	96.82	97.17	0.9700	33.293	96.11	63.57	0.7653	33.766	95.16	49.64	0.6525
0.25	43.702	97.04	95.66	0.9635	35.165	95.99	57.65	0.7203	33.333	95.17	43.55	0.5975
0.3	46.816	95.92	95.84	0.9588	35.829	95.80	46.42	0.6254	32.854	95.16	39.17	0.5549
0.35	45.548	96.47	96.18	0.9633	39.174	95.45	44.78	0.6096	37.612	93.18	32.71	0.4842
0.4	49.520	96.39	96.11	0.9625	41.317	96.10	41.96	0.5842	33.993	95.31	31.15	0.4695
0.45	54.242	96.38	95.96	0.9617	41.783	95.14	38.20	0.5451	37.144	94.43	27.19	0.4223
0.5	57.840	96.87	95.48	0.9617	44.485	94.83	34.32	0.5040	34.953	93.95	24.34	0.3867



(a) Non-linear dataset (b) Electrical dataset

Figure 18: MI search space on time delay

window pair. In contrast, it is difficult to find the target if we select wrong τ.

6.4 Effectiveness of Phase One

In this experiment, we compare two strategies in the first phase. As mentioned above, NLC$_E$ and NLC$_S$ work well for different distribution of correlated pairs. So, we use *correlated density* to measure the distribution of the correlated pairs. Formly, *correlated density* is the ratio between the sum of the lengths of all planted correlated window pairs and the total length n. The data generation process is the same as the non-linear dataset, but the length constraint is [300,700], and the delay constraint is [-30,30]. We generate ten different non-linear datasets with different correlated density.

The results of the experiments are shown in Table 5. It can be seen that NLC$_E$ always maintains good performance: precision and recall are always greater than 95%, and F1-score is always greater than 0.95. Regardless of the distribution of correlated window pairs, the performance of the window extending strategy in terms of data accuracy is almost unaffected. The tolerance of correlation density is a major advantage of this extending strategy. At the same time, the runtime increases as the density increases, since the number of windows needed to be searched also increases.

For window shrinking strategy, precision is hardly affected by the correlated density. However, recall and F1-score decreases as the density increases. This is determined by the nature of window shrinking strategy. In order to ensure the advantage of efficiency, it reduces the search space and inevitably misses some results. The greater the correlated density, the more correlated window pairs will be lost. It is more efficient than the extending strategy.

In summary, window extending strategy is more accurate and stable to variable data characteristics, but it takes more time. In contrast, window shrinking strategy has high precision and takes less time, but it will miss window pairs when the density increases. The choice between these two strategies is the consideration of accuracy and efficiency. If we are more focused on the accuracy or want to know the specific positions, and we are willing to spend more time, NLC$_E$ is a good choice. Otherwise, using window shrinking strategy is a better choice.

6.5 Case Study

Next, we compare them on three real-world datasets.

6.5.1 *Case 1: TE dataset.*

In this experiment, to make comparison fair, we determine the threshold as follows. We first sample data and set a PCC threshold, use a linear method to search results, based on which, we determine the MI threshold. Specifically, we sample 10 time series, the PCC threshold is set as 0.9, and MASS is used to detect correlations. There are 138 correlated window pairs found by MASS in total. We set the MI threshold to 0.8, there are 117 pairs with MI value greater than 0.8, which means that theoretically 85% of linear correlations can be found by methods using MI.

It can be seen from Table 6 that NLC$_E$ has found the most correlated window pairs ($|\mathcal{CP}|$), and the longest total correlation length ($\sum_{i=1}^{|\mathcal{CP}|} l_i$). TYCOS finds the second most $|\mathcal{CP}|$, but is much slower. MASS is fastest, but finds less correlated window pairs than NLC$_E$ and TYCOS. As examples in Figure 1, it is impossible for linear methods to identify such non-linear correlations. Jocor find least results, which is also due to the limitations of linear methods.

Table 6: TE dataset results

Methods	Runtime (s)	$\lvert\mathcal{CP}\rvert$	$\sum_{i=1}^{\lvert\mathcal{CP}\rvert} l_i$	$\bar{l}$
MASS	**597.79**	6114	945560	154.65
Jocor	1043.71	3297	536781	162.81
TYCOS	5226.97	6374	995260	156.14
NLC$_S$	1896.47	5585	1164839	208.57
NLC$_E$	2717.79	**7435**	**1635262**	**219.94**

6.5.2 *Case 2: stock market dataset.*

In this experiment, We compare MASS, Jocor, TYCOS and NLC. For NLC$_S$, w_e is set as 600 here. The threshold θ is determined similarly as TE dataset. Similarly, we sample 10 time series, set PCC threshold to 0.8, use MASS to find 74 linear correlated window pairs. We set MI threshold to 0.7, there are 70 pairs meeting the threshold condition, which can theoretically be detected by MI methods.

The result is shown in Table 7. NLC$_E$ and TYCOS find much more pairs ($|\mathcal{CP}|$) than Jocor and MASS, which still benefits from MI metric.

Compared with the synthetic datasets, TYCOS performs much better in stock market dataset. Due to the strong correlation of

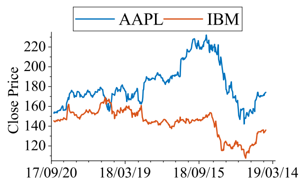
Table 7: Stock market dataset results

Methods	Runtime (s)	$\lvert\mathcal{CP}\rvert$	$\sum_{i=1}^{\lvert\mathcal{CP}\rvert} l_i$	$\bar{l}$
Jocor	4707.90	2398	704599	293.83
MASS	**457.75**	2526	742910	294.11
TYCOS	2483.38	3447	717370	208.11
NLC$_S$	805.59	2297	751481	327.16
NLC$_E$	1958.83	**4138**	**1368163**	**330.63**

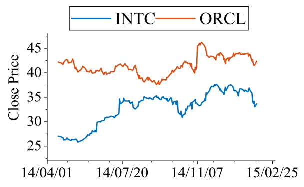
stocks in the same industry, imprecision of time delay has little effect. However, the average length found by TYCOS is the shortest. This is because TYCOS always stays in the starting window pair, the search process is stuck and cannot give better results. MASS finds more pairs than Jocor, and takes much less time. Although MASS is not proposed for the problem here, it has wide applicability and high efficiency.

Some detected non-linear windows are shown in Figure 19. For AAPL and IBM, the whole correlation length is 355 days and PCC value is 0.0965, of which the first 80 days show a positive correlation ($PCC = 0.7571$), the 81-200 days show a negative correlation ($PCC = -0.8493$), and the last 155 days show a positive correlation ($PCC = 0.8234$). For INTC and ORCL, the window correlation length is 200 days, of which the first 140 days show a negative correlation($PCC = -0.6256$), and the last 40 days show a positive correlation($PCC = 0.7272$). The overall PCC is close to 0, but $MI > 1$.

If positive and negative correlations alternate rapidly, the PCC value will be very low. However, we cannot deny that they are correlated. This change process is similar to the Cross relation in Table 3, and is very common in stock changes. MI can detect such alternating positive and negative correlations.



(a) AAPL VS IBM



(b) INTC VS ORCL

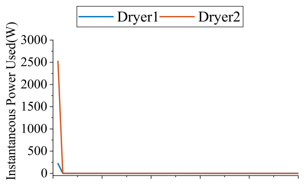
Figure 19: Non-linear pairs in stock market dataset

6.5.3 *Case 3: electrical dataset.*

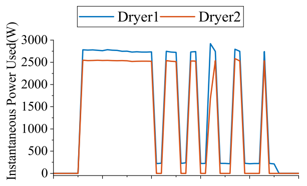
Electrical dataset records the instantaneous electrical power of various devices per minute. If the device is not in use, the power consumption is 0. We compare TYCOS and NLC. We set the initial threshold θ_0 to 0.2, and θ to 0.8.

MASS and Jocor are not compared here, since they cannot detect meaningful correlations in this dataset. They tend to detect the unused state of the device. As shown in Figure 20(a), analyzing the time series of two dryers, the PCC value of correlated window pairs found by MASS is approximately 1.0, but most results are meaningless. In contrast, the results of our approach are obviously more meaningful. Figure 20(b) shows an example.

The results are shown in Table 8. NLC$_E$ still finds the most pairs and has the longest length. TYCOS finds the second most pairs but takes much more time. NLC$_S$ with $w_e = 160$ is the fastest but finds fewer results.



(a) Found by linear method



(b) Found by non-linear method

Figure 20: Detected correlations in electrical dataset

Table 8: Electrical dataset results

Methods	Runtime (s)	$\lvert\mathcal{CP}\rvert$	$\sum_{i=1}^{\lvert\mathcal{CP}\rvert} l_i$	$\bar{l}$
TYCOS	37427.13	2871	162549	56.62
NLC$_S$ (180)	**850.57**	1504	122831	81.67
NLC$_S$ (240)	1107.27	1826	151943	**83.21**
NLC$_S$ (480)	854.90	1461	121321	83.04
NLC$_E$	2485.92	**4491**	**338053**	75.27

We analyze the results of NLC$_S$ by taking different values of w_e. When $w_e = 240$, $\lvert\mathcal{CP}\rvert$ is the most, and $\bar{l}$ is also the longest. But why $w_e = 240$ is better than $w_e = 180$ and $w_e = 480$? In the comparison experiment in section 6.4, shrinking strategy's precision is high but recall is low, which means many answers have been missed. In this real-world case, the distribution is uncertain. If w_e is set shorter ($w_e = 180$), the search range will become smaller since we should ensure that the window length is in [40,120]. The target may also be missed if it is just between two envelope windows. If w_e is set higher ($w_e = 480$), the search range will be wider. However, if the envelop window contains multiple correlations, other results would be missed. Therefore, w_e must be set at the suitable value to have a better performance.

We have quantified the coverage of results between NLC$_E$ and TYCOS from two directions. First, we mark all $\mathcal{CP}$ found by TYCOS, check with $\mathcal{CP}$ found by NLC$_E$ in these marked sequences, and count the overlap length ratio. The statistical coverage in the other direction is also the same way. The results are shown in Table 9. Most pairs found by TYCOS can be found by NLC$_E$, while most of pairs found by NLC$_E$ cannot be found by TYCOS.

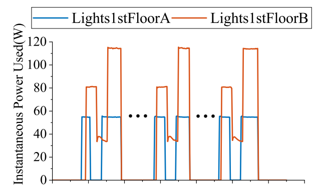
Table 9: Coverage results between NLC$_E$ and TYCOS

Types	X Coverage	Y Coverage	Total Coverage
NLC$_E$->TYCOS	0.8758	0.8931	0.8845
TYCOS->NLC$_E$	0.3700	0.3657	0.3678

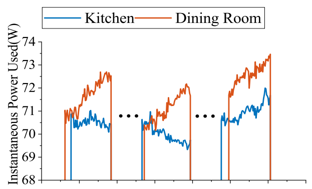
We show examples of non-linear pairs in Figure 21. *Lights1stFloorA* refers to a subset of lighting on first floor, including lights living room, hallway, and bedroom 4, while *Lights1stFloorB* includes hallway, bathroom, kitchen, and dining room. We have found a total of 60 correlated window pairs between *Lights1stFloorA* and *Lights1stFloorB*, of which 40 pairs' PCC is less than 0.5. Between *Kitchen* and *Dining room*, 42 correlated window pairs are found, of which 13 pairs' PCC is less than 0.5.

7 RELATED WORKS

Past research usually uses covariance and pearson correlation coefficient (PCC) [20] to measure the correlation between data. In the



(a) Lighting system correlations



(b) Kitchen VS Dining room

Figure 21: Non-linear pairs in electrical dataset

financial field, Buraschi *et al.* [2] study economically significant covariance components and derive optimal portfolio implications for economies. In the medical field, Huang *et al.* [12] apply PCC to identify disease-specific biomarkers by comparing the gene expression profiles between normal and disease states. In the field of geology, Dean *et al.* [5] analyze correlation to study the stratigraphic sequence.

Recently, researchers expand the concept of correlation from the basic definition. Sarma *et al.* [24] and Alawini *et al.* [1] capture relatedness between data tables. Roy *et al.* [23] explain relations among the outputs of SQL queries based on the notion of intervention. Pochampally *et al.* [21] model correlations between sources and apply it in truth finding.

However, these studies all focus on the overall correlation and lack of attention to the local time windows. Also, there exist some works searching the local correlation. Rakthanmanon *et al.* [22] propose MASS to quickly find the most similar subsequence to the query in the time series, which greatly improves the search performance. Unfortunately, MASS cannot perform the correlation search between two long time series when the position of correlation is uncertain since the search relies on queries. Mueen *et al.* [18] propose Jocor, an efficient algorithm to join two long time series in their most correlated segments of arbitrary lag and duration. Jocor also takes into account the influence of window length and use length-adjusted distance. Yeh *et al.* [27] propose MatrixProfile series algorithms to search for time series subsequence all-pairs-similarity-search. Although the time window is considered more carefully, the length of the window of MatrixProfile is still fixed.

All approaches mentioned above use traditional correlation, which fundamentally determines that they are limited to linear and monotonic dependencies, and unable to detect complex correlation correlations. Ho *et al.* [11] study the use of MI for correlation discovery, and proposed AMIC to search for the multi-scale correlation of big data. Later, Ho *et al.*propose the complete TYCOS approach in [9], which considers the time delay factor and improves the performance. However, TYCOS suffers from dealing with the large time delay. Also the efficiency is much worse the our approach.

8 CONCLUSION

In this paper, we study the problem of non-linear correlation search in two long time series. We propose a two-phase approach. In phase one, we propose two strategies to generate candidates. Window shrinking strategy is suitable for the sparsely distributed correlations, while window extending strategy is slightly slower, but can find more pairs when the correlations are distributed densely. In phase two, we propose a nested search strategy to refine candidates. Experimental results verify the efficiency and effectiveness of NLC.

REFERENCES

[1] Abdussalam Alawini, David Maier, Kristin Tufte, and Bill Howe. 2014. Helping scientists reconnect their datasets. In *Proceedings of the 26th International Conference on Scientific and Statistical Database Management*. 1–12.
[2] Andrea Buraschi, Paolo Porchia, and Fabio Trojani. 2010. Correlation risk and optimal portfolio choice. *The Journal of Finance* 65, 1 (2010), 393–420.
[3] Edmund K Burke and Yuri Bykov. 2017. The late acceptance hill-climbing heuristic. *European Journal of Operational Research* 258, 1 (2017), 70–78.
[4] C. Chatfield. 2004. *The analysis of time series: an introduction*. Chapman & Hall/CRC.
[5] Walter E Dean and Roger Y Anderson. 1974. Application of some correlation coefficient techniques to time-series analysis. *Journal of the International Association for Mathematical Geology* 6, 4 (1974), 363–372.
[6] James J Downs and Ernest F Vogel. 1993. A plant-wide industrial process control problem. *Computers & chemical engineering* 17, 3 (1993), 245–255.
[7] Liying Fang, Han Zhao, Pu Wang, Mingwei Yu, Jianzhuo Yan, Wenshuai Cheng, and Peiyu Chen. 2015. Feature selection method based on mutual information and class separability for dimension reduction in multidimensional time series for clinical data. *Biomedical Signal Processing and Control* 21 (2015), 82–89.
[8] William Healy, Farhad Omar, Lisa Ng, Tania Ullah, William Payne, Brian Dougherty, and A Hunter Fanney. 2016. Net zero energy residential test facility instrumented data; year 2. *National Institute of Standards and Technology, Gaithersburg, MD* (2016).
[9] Nguyen Ho, Torben Bach Pedersen, Van Long Ho, and Mai Vu. 2020. Efficient Search for Multi-Scale Time Delay Correlations in Big Time Series Data.. In *EDBT*. 37–48.
[10] Nguyen Ho, Torben Bach Pedersen, Mai Vu, Christophe AN Biscio, et al. 2019. Efficient bottom-up discovery of multi-scale time series correlations using mutual information. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 1734–1737.
[11] Nguyen Ho, Huy Vo, Mai Vu, and Torben Bach Pedersen. 2019. Amic: An adaptive information theoretic method to identify multi-scale temporal correlations in big time series data. *IEEE Transactions on Big Data* 7, 1 (2019), 128–146.
[12] Hung-Chung Huang, Siyuan Zheng, and Zhongming Zhao. 2010. Application of Pearson correlation coefficient (PCC) and Kolmogorov-Smirnov distance (KSD) metrics to identify disease-specific biomarker genes. *Bmc Bioinformatics* 11, 4 (2010), 1–2.
[13] S. K. Jensen, T. B. Pedersen, and C. Thomsen. 2017. Time Series Management Systems: A Survey. *TKDE* 29, 11 (2017), 2581–2600.
[14] Donald R Jones, Cary D Perttunen, and Bruce E Stuckman. 1993. Lipschitzian optimization without the Lipschitz constant. *Journal of optimization Theory and Applications* 79, 1 (1993), 157–181.
[15] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. 2004. Estimating mutual information. *Physical review E* 69, 6 (2004), 066138.
[16] Yuhong Li, Leong Hou U, Man Lung Yiu, and Zhiguo Gong. 2013. Discovering Longest-lasting Correlation in Sequence Databases. *Proc. VLDB Endow.* 6, 14 (2013), 1666–1677.
[17] Matthieu Lucke, Xueyu Mei, Anna Stief, Moncef Chioua, and Nina F Thornhill. 2019. Variable selection for fault detection and identification based on mutual information of alarm series. *IFAC-PapersOnLine* 52, 1 (2019), 673–678.
[18] Abdullah Mueen, Hossein Hamooni, and Trilce Estrada. 2014. Time series join on subsequence correlation. In *2014 IEEE International Conference on Data Mining*. IEEE, 450–459.
[19] Oleh Onyshchak. 2020. Stock Market Dataset. https://doi.org/10.34740/KAGGLE/DSV/1054465
[20] Karl Pearson. 1895. VII. Note on regression and inheritance in the case of two parents. *proceedings of the royal society of London* 58, 347-352 (1895), 240–242.
[21] Ravali Pochampally, Anish Das Sarma, Xin Luna Dong, Alexandra Meliou, and Divesh Srivastava. 2014. Fusing data with correlations. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 433–444.
[22] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. 2012. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 262–270.
[23] Sudeepa Roy and Dan Suciu. 2014. A formal approach to finding explanations for database queries. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 1579–1590.
[24] Anish Das Sarma, Lujun Fang, Nitin Gupta, Alon Y Halevy, Hongrae Lee, Fei Wu, Reynold Xin, and Cong Yu. 2012. Finding related tables. (2012).
[25] Jiaye Wu, Yang Wang, Peng Wang, Jian Pei, and Wei Wang. 2018. Finding maximal significant linear representation between long time series. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1320–1325.

[26] Chunhua Yang, Sen Xie, Xiaofeng Yuan, Xiaoli Wang, and Yongfang Xie. 2018. A new data reconciliation strategy based on mutual information for industrial processes. *Industrial & Engineering Chemistry Research* 57, 38 (2018), 12861–12870.

[27] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. 2016. Matrix profile I: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets. In *2016 IEEE 16th international conference on data mining (ICDM)*. Ieee, 1317–1322.