Robust Time Series Chain Discovery with Incremental Nearest Neighbors

Li Zhang\textsuperscript{*8}, Yan Zhu\textsuperscript{18}, Yifeng Gao\textsuperscript{†}, Jessica Lin\textsuperscript{*}

\textsuperscript{*}Department of Computer Science, George Mason University, USA
\textsuperscript{†}Google, USA

\textsuperscript{8}Department of Computer Science, University of Texas Rio Grande Valley, USA
{\{lzhang18,jessica\}@gmu.edu, zhuyan@google.com, yifeng.gao@utrgv.edu}

Abstract—Time series motif discovery has been a fundamental task to identify meaningful repeated patterns in time series. Recently, time series chains (TSCs) were introduced as an expansion of time series motifs to identify the continuous evolving patterns in time series data. TSCs are shown to be able to reveal latent continuous evolving trends in the time series, and identify precursors of unusual events in complex systems. However, existing TSC definitions lack the ability to accurately cover the evolving part of a time series: the discovered chains can be easily cut by noise and can include non-evolving patterns, making them impractical in real-world applications. In this work, we introduce a new TSC definition based on an incremental nearest neighbor concept which can better locate the evolving patterns while excluding the non-evolving ones, and propose two new quality metrics to rank the discovered chains. With extensive empirical evaluations, we demonstrate that the proposed TSC definition is significantly more robust to noise than the state of the art, and the top ranked chains discovered can reveal meaningful regularities in a variety of real world datasets.

Index Terms—time series, time series chains, drift, prognostics

I. INTRODUCTION

In the last two decades, the task of finding repetitive patterns in time series data, known as motif discovery, has received a lot of attentions in the research community due to its wide range of applications across many different domains [1]–[4]. Recently, a new primitive called the time series chain (TSC) was introduced as a new tool to capture the time series data over time [5], [6]. Informally, a time series chain is an ordered set of subsequences extracted from a time series, where adjacent subsequences in the chain are similar, but the first and the last are arbitrarily dissimilar. Time series chains can capture any potential drift accumulated over time, which widely exists in many complex systems, natural phenomena and societal changes [5]–[7].

A typical application of time series chains is prognostics. As noted by previous studies [7], [8], “most equipment failures are preceded by certain signs, conditions, or indications that such a failure was going to occur.” Time series chains can identify not only a single precursor, but also a whole sequence of patterns revealing the continuous and gradual change of the system, helping analyzers uncover the reasons at an early stage and prevent catastrophic failures.

The concept of chains was first introduced in [5] (TSC17) based on a bi-directional definition, and later on [6] (TSC20)\textsuperscript{8}.

\textsuperscript{8}equal contribution

Fig. 1. Example chains in the two-dimensional space where each dot represents a subsequence in the chain. (a) A clean chain that can be discovered by both TSC17 and TSC20. (b) A chain with a single deviated node (Node 5) that would be missed by TSC17. (c) A chain that shows an evolving trend but includes more fluctuations. Neither TSC17 nor TSC20 can find this chain. (d) A chain (the red nodes) drifting from a steady state (the blue nodes). TSC20 would attach the blue nodes to the discovered chain, which is undesirable.

To address these limitations, in this work we propose a novel time series chain definition TSC22, which exploits an idea to track how the nearest neighbor of a pattern changes over time to improve the robustness of chain discovery. Similar to TSC17, TSC22 is a bi-directional definition but with much more relaxed constraints, and it does not rely on any angle constraint as in TSC20. Furthermore, we propose a new quality metric to rank the chains discovered by our method. With extensive empirical evaluations, we demonstrate that the proposed method is much more robust to noise than the state of the art, and the top ranked chains discovered can reveal meaningful regularities in a variety of real world datasets.
II. BACKGROUND AND RELATED WORK

The review is brief as the time series chain is a relatively new topic. The closest related works are the ones by Zhu et al. [5] and Imamura et al. [6]. Zhu et al. [5] first introduce the concept of time series chains. The work enforces every adjacent pair of subsequences in the chain to be the left and right nearest neighbors of each other, and reports the longest chain as the top chain in the time series. Although the concept is simple and intuitive, the bi-directional nearest neighbor requirement is shown to be too strict in many real-world applications, as the chain can be easily cut by data fluctuations and noise [6]. Imamura et al. [6] relax the bi-directional constraint by only enforcing the single-directional left nearest neighbor requirement, and add a pre-defined angle constraint to guarantee the directionality of the discovered chains. The concept is shown to be more robust than [5] in some conditions, but a new angle parameter is introduced, and we have observed that in some real-world scenarios, no meaningful chains can be found no matter how we set this parameter. We will elaborate more on the theoretical limitations of these two chain definitions in Section IV.

Other works on time series chains have different problem settings. Wang et al. [10] explore methods to speed up chain discovery in streaming data using the existing time series chain definition [5]. Zhang et al. [11] propose a method to detect joint time series chains across two time series, while we focus on improving the chain definition in a single time series.

III. DEFINITIONS

In this section, we first review necessary time series notations, then consider the formal definition of time series chains.

A. Time Series Notations

We start with fundamental time series definitions.

Definition 1. A time series $T = [t_1, t_2, \ldots, t_n]$ is an ordered list of data points, where $t_i$ (1 ≤ $i$ ≤ $n$) is a finite real number and $n$ is the length of time series $T$.

Definition 2. A time series subsequence $S_i = [t_i, t_{i+1}, \ldots, t_{i+l-1}]$ is a contiguous set of points in time series $T$ starting from position $i$ with length $l$. Typically $l < n$, and 1 ≤ $i$ ≤ $n - l + 1$.

Given a query subsequence, we can compute its distance to every subsequence before (or after) it in time series $T$. We call this a left distance profile (or right distance profile) [5].

Definition 3. A left distance profile $DL_i$ of time series $T$ is a vector containing the Euclidean distances between a given subsequence $S_i \in T$ and every subsequence before $S_i$. Formally, $DL_i = [d(S_i, S_1), d(S_i, S_2), \ldots, d(S_i, S_{i-1/2})]$.

Definition 4. A right distance profile $DR_i$ is a vector where $DR_i = [d(S_i, S_{i+1/2}), \ldots, d(S_i, S_{n-1})]$.

Following the original Matrix Profile works [1], [3], here we use the z-normalized Euclidean distance instead of the Euclidean distance to achieve scale and offset invariance [12].

We can easily find the left nearest neighbor (LNN) and the right nearest neighbor (RNN) of a subsequence by examining the minimum values in the left/right distance profiles. We store their information in the left/right matrix profiles.

Definition 5. A left matrix profile $MP_L(i)$ is a two dimensional vector of size $2 \times (n - l + 1)$ where $MP_L(1, i) = \min(DL_i)$ and $MP_L(2, i) = \arg\min(DL_i)$.

Definition 6. A right matrix profile $MP_R(i)$ is a two dimensional vector of size $2 \times (n - l + 1)$ where $MP_R(1, i) = \min(DR_i)$ and $MP_R(2, i) = \arg\min(DR_i)$.

B. Existing Time Series Chain Definitions

Existing time series chain definitions [5] [6] are both developed upon the backward time series chain, which can be easily obtained given the left matrix profile.

Definition 7. A backward chain $TSC_{BWD}$ of time series $T$ is a finite ordered set of time series subsequences: $TSC_{BWD} = [S_{C_1}, S_{C_2}, S_{C_3}, \ldots, S_{C_m}]$, where $C_1 > C_2 > \ldots > C_m$ are indices in time series $T$, such that for any $1 \leq i < m$, we have $LNN(S_{C_i}) = S_{C_{i+1}}$.

For clarity, we denote a subsequence in a time series chain as a node. We call the first node in the backward chain the start node and the last node the end node. Here in $TSC_{BWD}$, $S_{C_1}$ is start node and $S_{C_m}$ is the end node.

Analogously, we can obtain the forward time series chain from the right matrix profile.

Definition 8. A forward chain $TSC_{FWD}$ of time series $T$ is a finite ordered set of time series subsequences: $TSC_{FWD} = [S_{C_1}, S_{C_2}, S_{C_3}, \ldots, S_{C_m}]$ where $C_1 < C_2 < \ldots < C_m$ are indices in time series $T$, such that for any $1 \leq i < m$, we have $RNN(S_{C_i}) = S_{C_{i+1}}$.

Existing works [5] [6] define chains by placing different constraints on the the backward chain to guarantee its directionality. [5] requires that a time series chain must be both a backward chain and a forward chain. We call this a bi-directional time series chain.

Definition 9. A bi-directional time series chain ($TSC_{17}$) of a time series $T$ is a finite ordered set of time series subsequences: $TSC = [S_{C_1}, S_{C_2}, S_{C_3}, \ldots, S_{C_m}]$ where $C_1 > C_2 > \ldots > C_m$, such that for any $1 \leq i < m$, we have $LNN(S_{C_i}) = S_{C_{i+1}}$ and $RNN(S_{C_{i+1}}) = S_{C_i}$.

Here we denote the size of the set $m$ as the length of the chain. Fig. 2 shows how TSC17 works. Assume that the subsequence length is 1, and we use the absolute difference between the numbers to measure their distance. Starting from 5, we check whether the LNN and the RNN of each node can form a loop. The extracted chain is $5 \leftrightarrow 4 \leftrightarrow 3.3 \leftrightarrow 3 \leftrightarrow 2 \leftrightarrow 1$.

In [6], a time series chain is defined by placing an angle constraint on the backward chain. We call this a geometric time series chain.
Definition 10. A geometric time series chain (TSC20) of a time series $T$ is a finite ordered set of time series subsequences: $TSC = \{S_{C_1}, S_{C_2}, S_{C_3}, \ldots, S_{C_m}\}$ ($C_1 > C_2 > \ldots > C_m$) such that for any $1 \leq i < m$, we have $LNN(S_{C_i}) = S_{C_{i+1}}$, and for any $2 \leq i < m$, we have the $i$-th angle $\theta_i \leq \theta$, where $\theta_i = \cos^{-1}\left(\frac{S_{C_i} - S_{C_{i+1}}}{\|S_{C_i} - S_{C_{i+1}}\|, \|S_{C_{i+1}} - S_{C_{i+1}}\|}\right)$, and $\theta$ is a predefined threshold.

Fig. 3. TSC20 places an angle constraint on the backward chain.

Fig. 4. A flip of the subsequences can easily break a bi-directional chain.

Fig. 5. (top) A chain that evolves in a zig-zag pattern due to noise in the data. (bottom) TSC20 fails to find the chain as the first angle is too large.

B. The weakness of TSC20

The geometric chain definition in TSC20 [6] replaces the bi-directional constraint of TSC17 with an angle constraint to be more tolerant of noise. However, we found that the angle constraint can bring in new problems, causing TSC20 to miss obvious chains in some situations, while in some others attaching unwanted patterns to the discovered chain.

1) Failure to detect obvious chains: Consider the 2-dimensional example in Fig. 5.top. Assume that the subsequences show up in the following order in time: $S_9, S_8, S_7, \ldots, S_1$. Although the subsequences evolve in a zig-zag pattern, they show a clear trend moving from left to right. However, TSC20 fails to detect the chain. In Fig. 5.bottom, we can see the first angle in the backward chain $\theta_1 = 90$ degrees, much larger than the suggested threshold 40 degrees in [6] and the chain breaks instantly. We can repeat the process by resetting the anchor to the next subsequence, but the angles are all larger than the threshold, and no chain can be found.

2) Attaching unwanted patterns to the chain: In Fig. 6.top, the subsequences are first in a steady state (shown in blue), then they start to drift all the way to the right (shown in red). Ideally, we want the chain discovery algorithm to find all the red nodes, but not the blue ones, so that the first red node on the left can tell us exactly when the system starts to drift (i.e., the change point). This is critical especially in the prognostics use case, as the time information can help us identify the root cause of the drift. However, as shown in Fig. 6.middle, all the consecutive nodes on the backward chain meet the angle constraint of TSC20; the red nodes are evolving in a quasi-linear trend, so the direction angle is close to zero. The blue nodes (shown in Fig. 6.bottom) also form very small angles as they are very close to each other while far away from the anchor. As a result, TSC20 cannot remove the blue nodes from the chain, failing to detect the change point.

V. OUR PROPOSED METHOD

To address the aforementioned limitations in existing TSC definitions, we propose a new time series chain discovery method named TSC22. In this section, we first introduce our new chain definition, then discuss how we rank the chains discovered by this definition.
A. Incremental Nearest Neighbors

Our chain discovery method leverages an idea from [13] to trace how the nearest neighbor of a time series sequence changes over time. We use the running 1-dimensional example in Fig. 7 to show how this works. Here we scan from the end of the time series all the way left to subsequence 2, and record how the nearest neighbor (based on absolute value difference) of subsequence 2 gets updated: 5, 4, 3. We call the set \{3, 4, 5\} an Incremental Nearest Neighbor Set (INNS).

**Definition 11.** The Incremental Nearest Neighbor Set (INNS) of a subsequence \(S_i\) is a set of subsequences \(S_j\) (\(i < j \leq n - l + 1\)): \(\{S_j\mid d(S_i, S_j) < d(S_i, S_k)\forall k : j < k \leq n - l + 1\}\).

Note that the right nearest neighbor (RNN) of a subsequence always belongs to the INNS of that subsequence.

B. Finding All TSCs with Incremental Nearest Neighbors

Based on the Incremental Nearest Neighbors, we introduce a key component in our chain definition, the critical nodes:

**Definition 12.** A time series subsequences \(S_i\) of a time series \(T\) is a Critical Node (CN) if \(S_i \in \text{INNS}(LNN(S_i))\).

We can find all the critical nodes in a time series \(T\); we call this a critical node set (denoted as \(CN_T\)). With this, we formally define TSC22, a relaxed bi-directional TSC:

**Definition 13.** A relaxed-bi-directional time series chain (TSC22) of a time series \(T\) is a finite ordered set of time series subsequences: \(TSC = [S_{C_1}, S_{C_2}, S_{C_3}, \ldots, S_{C_m}]\) \((C_1 > C_2 > \ldots > C_m)\), such that:

1. for any \(1 \leq i < m\), we have \(LNN(S_{C_i}) = S_{C_{i+1}}\), and
2. \(S_{C_1} \in CN_T\), and
3. for any \(1 < i \leq m\), we have \(S_{C_j} \in \text{INNS}(S_{C_i})\), where \(j = \arg\max_{1 \leq j < i} S_{C_j} \in CN_T\).

Similar to TSC17 and TSC20, TSC22 is built on top of a backward chain. We call it a “relaxed” bi-directional chain as our restrictions on the forward direction is not as strict as that of TSC17: we rely on the \(INNS\) instead of a single \(RNN\).

Fig. 8 shows a one-dimensional time series example of how TSC22 works. Based on (1) and (2), we can see that TSC22 is a backward chain which starts from a critical node. Let us try to develop a chain from node 5. According to Definition 12, the critical nodes on the backward chain are 5, 4, and 2.

Condition (3) requires that for every node \(S_{C_i}\) in the chain (except for the starting node), the closest critical node in the chain after \(S_{C_i}\) must be an element in \(INNS(S_{C_i})\). By checking on every node (starting from the second node) in the backward chain, we can see that 4, 3, 2, and 1 do meet this condition, but 20 does not, as \(1 \notin \text{INNC}(20) = \{23, 34, 5\}\), so the chain breaks here and the generated chain is \(5 \rightarrow 4 \rightarrow 3.3 \rightarrow 2 \rightarrow 1\). Essentially, condition (3) ensures the directionality of the generated chain. As shown in Fig. 8, it makes sure that subsequence 2 is closer to subsequence 1 than every subsequence after 2, 4 is closer to 2 and 3.3 than every subsequence after 4, etc.

By comparing Fig. 8 with Fig. 4, we can see that TSC22 allows us to extract a meaningful chain, even when part of the patterns in the chain are flipped. The reader may verify that the zig-zag chain in Fig. 5 also satisfies our definition. This shows that TSC22 is more robust than both TSC17 and TSC20 in the face of noisy data. We will further demonstrate this claim with high-dimensional data in Section VI.

Algorithm 1 finds all the TSC22 chains in a time series. We first leverage STUMP [13] to compute the Matrix Profile that contains the \(LNN\) and \(INNS\) information for every subsequence, then we find all the critical nodes in the time series and grow chains from them. Each critical node is only visited once. We store the sub-chains corresponding to each discovered chain in the our results. The time and space complexity of our algorithm is identical to that of TSC20 [6].
C. Ranking

We use two quality metrics to rank the discovered chains. Effective Length inspired by TSC20 [6], we compute an effective length metric $L_{eff}$, which measures both divergence and graduality at the same time:

$$L_{eff} = \left( \frac{d(S_{C_i}, S_{C_m})}{\max_{1 \leq i \leq m-1} d(S_{C_i}, S_{C_{i+1}})} \right)$$

where $\lceil . \rceil$ denotes rounding to the nearest integer. The numerator is the distance between the first node and the last node in the chain, and the denominator is the maximum distance between all pairs of consecutive nodes in the chain. The metric essentially tells us the minimum number of steps possible to reach the end node from the start node if we are moving in a linear trace. As $L_{eff}$ is an integer, there will be ties. We first rank the chains by their $L_{eff}$ scores, choose the maximum, then do a fine-grain ranking based on the correlation length. Correlation Length $L_{corr}$ is computed as follows:

$$L_{corr} = \sum_{i=1}^{m-1} |Corr(S_{C_i}, S_{C_{i+1}})Corr(S_{C_i}, S_{C_{i+1}})|$$

where $Corr(\cdot) \in [-1, 1]$ is the Pearson Correlation Coefficient of the z-normalized subsequences. The correlation length prefers long chains with similar consecutive subsequences.

VI. EMPIRICAL EVALUATION

In this section, we demonstrate that the proposed method is more robust than the current state-of-the-art TSC discovery methods, TSC17 (ICDM’17) [5] and TSC20 (KDD’20) [6] on real-world data. To ensure fair comparison, we use the original source code of both methods in Matlab, and we use the default direction angle for TSC20. To make the results easily reproducible, we built a supporting webpage [14] that contains all the data and code used in this section, as well as more experimental results on a large-scale synthetic dataset.

A. Case Study: Robustness Analysis on Penguin Activity Data

![Fig. 9. Chains detected in a clean penguin activity time series as the penguin dives into the water. TSC22 is able to find a high-quality chain that covers the whole diving range; TSC20 misses a few nodes at the end and TSC17 misses nodes on both sides.](image)

We start with the penguin activity time series in [5], which records the x-axis acceleration of the bird as it moves. Fig. 9 shows a 22.5s snippet of the data (recorded at 40Hz) as the penguin dives into the water. The bird reaches the water at around 2 seconds, and then the depth starts to increase. The subsequence length is 25. As shown in Fig. 9, TSC22 finds a high-quality chain with a clear evolving trend: at the beginning the first peak in the pattern is slightly lower than the second peak, then over time the first peak grows higher and the second peak becomes lower. This chain indicates how the penguin adjusts its flapping to balance between the buoyancy and the increasing water pressure [5]. The chain found by TSC22 covers the full diving range, while the TSC20 chain misses a few nodes at the end and the TSC17 chain misses a few more on both sides. Though the chains found by TSC20 and TSC17 can still show the evolving trend of the patterns, they fail to indicate the precise start/end time of the dive.

Since we are reporting the top-1 chain, the reader may wonder whether it is the definition or the ranking mechanism that prevents TSC17 and TSC20 from finding the TSC22 chain in Fig. 9. The answer is the former. Similar as in Fig. 4, most of the consecutive nodes on the chain do not satisfy the bi-directional nearest neighbor constraint of TSC17 due to the natural fluctuations in the data (e.g., the penguin may have changed its moving direction to avoid other animals). And as we trace the TSC22 chain backwards, we find that the directional angle formed by the first three nodes (recall Fig. 5) is 114 degrees, which is much higher than the TSC20 recommended threshold (40 degrees), so the chain breaks instantly. This example verifies the weaknesses of TSC17 and TSC20 in high-dimensional, real-world data with natural fluctuations and demonstrates the robustness of TSC22.

B. Case Study: Change Point Detection with Tilt Table Data

In this section, we use the tilt table data in [5] to compare how the three methods in consideration perform when a system transitions from a steady state to a drifting state. Fig. 10 shows the the arterial blood pressure (ABP) signal of a patient lying on a tilt table. Here we use a subsequence length of 180, roughly the length of a cardiac cycle. The table is flat at the beginning, and then starts to tilt. We would expect chains to be detected after the the tilt and no chain should appear before the tilt. We also expect a detected chain not to cross the tilt/non-tilt boundary, as the repeated patterns before the tilt would deteriorate the purity and interpretability of the chain.

![Fig. 10. The ABP time series of a patient in a tilt table experiment. The table is first flat, then it starts to tilt. The top chains detected by TSC22 and TSC17 only include the evolving patterns after the tilt, while TSC20 detects a chain that spans across the boundary, failing to locate the change point.](image)

Note that this kind of data is very typical in many other domains, especially in prognostics, where a system operates at a stable state at the beginning, and then start to deteriorate.
It is essential to correctly locate the change point (i.e., the beginning of the drift), so people can use that information to identify implicit mechanisms that could have led to the drift, preventing systematic failures at an early stage.

In Fig. 10, we can see that both TSC22 and TSC17 successfully discover chains that cover only the tilting section, while TSC20 detects a chain that spans across the tilting section and the steady section. It is not hard to understand why TSC20 fails to produce a pure chain in this case (recall Fig. 6): the direction angle formed between any pair of subsequences before the tilt and the anchor point would be close to zero, as these subsequences are almost identical. The single-directional nature of TSC20 simply cannot stop the backward chain from growing into the steady section. In contrast, the bi-directional restrictions in both TSC22 and TSC17 precisely break the chain right at the change point.

C. Case Study: Finding Chains in Human Gait Force Data

![Fig. 11. A human gait force time series collected on a threadmill with increasing speeds. TSC22 discovers a chain that spans across all four different speed zones, while the TSC20 chain only covers two and TSC17 only one.](image)

We next evaluate the chain discovery methods on a human gait force time series. As shown in Fig. 11, the time series represents the posterior-anterior direction force detected by the sensor on a split-belt treadmill, which operates at four different running speeds: 0.6, 0.7, 0.8, and 0.9 m/s. Here we use a subsequence length of 100, close to a cycle in the data. We can see from Fig. 11 that the TSC22 chain spans over all different speed zones. The sharpening dips in the chain patterns indicate that as the speed increases, the force changes more quickly, and the participant’s foot spend less time on the threadmill floor. The shortening period between the dip and the peak also indicate that the running pace becomes faster. However, despite the fact that the patterns are evolving across the whole time series, TSC20 and TSC17 can only detect chains on a small portion of the data, failing to cover the complete range. Why do TSC20 and TSC17 fail in this case? Note that as the speed is increasing at a piecewise-constant fashion, the patterns within the same speed zone are relatively similar to each other, evolving at a much smaller pace. As the participant’s movements naturally include fluctuations, a lot of patterns get flipped (discussed in Sec. IV.A) within a speed zone, and the overall evolving trace is similar to the zig-zag chains shown in Fig. 1.c and Fig. 5 instead of the one in Fig. 1.a. Note that the effect of the fluctuations is even more severe now as we are facing high-dimensional data. As a sanity check, we find that none of the nodes on the TSC22 chain in Fig. 11 meet the bi-directional nearest neighbor constraint of TSC20, and the direction angle formed by the last three nodes on the chain (i.e., the first three nodes in the backward chain) is 86 degrees, much larger than the angle limit of 40 degrees in TSC20. This example further demonstrates the superior robustness of TSC22 on noisy real-world time series data.

VII. CONCLUSION

In this work, we propose a novel TSC definition TSC22, which exploits an idea to track how the nearest neighbor of a pattern changes over time to improve the robustness of the chain against noise. In addition, two new quality metrics are proposed to effectively rank the detect chains. Experiments show that the new definition is much more robust than the state-of-the-art TSC definitions, and the top ranked chain discovered by our method can reveal meaningful regularities in a variety of applications under different noisy conditions.

REFERENCES

[1] Y. Zhu, Z. Zimmerman, N. S. Senobari, C.-C. M. Yeh, G. Funning, A. Mueen, P. Brisk, and E. Keogh, “Matrix profile ii: Exploiting a novel algorithm and graph to break the one hundred million barrier for time series motifs and joins,” in Data Mining (ICDM), 2016 IEEE 16th International Conference on. IEEE, 2016, pp. 739–748.

[2] L. Zhang, Y. Gao, and J. Lin, “Semantic discord: Finding unusual local patterns for time series,” in Proceedings of the 2020 SIAM International Conference on Data Mining. SIAM, 2020, pp. 136–144.

[3] C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. Keogh, “Matrix profile i: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets,” in 2016 IEEE 15th international conference on data mining (ICDM). Ieee, 2016, pp. 1317–1322.

[4] Y. Gao, Q. Li, X. Li, J. Lin, and H. Rangwala, “Trajviz: a tool for visualizing patterns and anomalies in trajectory,” in ECML-PKDD. Springer, 2017, pp. 428–431.

[5] Y. Zhu, M. Imamura, D. Nikovski, and E. Keogh, “Matrix profile vii: Time series chains: A new primitive for time series data mining (best student paper award),” in 2017 IEEE International Conference on Data Mining (ICDM). IEEE, 2017, pp. 695–704.

[6] M. Imamura, T. Nakamura, and E. Keogh, “Matrix profile xxi: A geometric approach to time series chain improves robustness,” in Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 1114–1122.

[7] H. P. Bloch and F. K. Geitner, Practical Machinery Management for Process Plants: Volume 2: Machinery Failure Analysis and Troubleshooting. Elsevier, 1997.

[8] R. Ahmad and S. Kamarudin, “An overview of time-based and condition-based maintenance in industrial application,” Computers & industrial engineering, vol. 63, no. 1, pp. 135–149, 2012.

[9] A. K. Jardine, D. Lin, and D. Banjevic, “A review on machinery diagnostics and prognostics implementing condition-based maintenance,” Mechanical systems and signal processing, vol. 20, no. 7, pp. 1483–1510, 2006.

[10] S. Wang, Y. Yuan, and H. Li, “Discovering all-chain set in streaming time series,” in PAKDD. Springer, 2019, pp. 306–318.

[11] L. Zhang, N. Patel, X. Li, and J. Lin, “Joint time series chain: Detecting unusual evolving trend across time series,” in Proceedings of the 2022 SIAM International Conference on Data Mining (SDM). SIAM, 2022, pp. 208–216.

[12] J. Lin, E. Keogh, L. Wei, and S. Lonardi, “Experiencing sax: a novel symbolic representation of time series,” Data Mining and knowledge discovery, vol. 15, no. 2, pp. 107–144, 2007.

[13] Y. Zhu, C.-C. M. Yeh, Z. Zimmerman, and E. Keogh, “Matrix profile xvi: Indexing the matrix profile to allow arbitrary range queries,” in 2020 IEEE 36th International Conference on Data Engineering (ICDE), 2020, pp. 1846–1849.

[14] Supporting webpage. [Online]. Available: https://sites.google.com/view/robust-time-series-chain-22