

Article

Efficient Time Series Visual Exploration for Insight Discovery

Heba Helal ^{1,2}  and Mohamed A. Sharaf ^{1,*}¹ Department of Computer Science and Software Engineering, College of Information Technology, United Arab Emirates University, Al Ain 15551, United Arab Emirates; 202090034@uaeu.ac.ae² Computer Engineering and Systems, Mansoura University, Al Mansurah 35516, Egypt

* Correspondence: msharaf@uaeu.ac.ae

Abstract

Visual exploration of time series data is essential for uncovering meaningful insights in domains such as healthcare monitoring and financial analysis, yet it remains computationally challenging due to the combinatorial explosion of potential subsequence comparisons. For long time series, an exhaustive comparison of all possible subsequence pairs becomes prohibitively expensive, limiting interactive exploration. This paper presents the TiVEx (Time Series Visual Exploration) family of algorithms for efficiently discovering the top- k most dissimilar subsequence pairs in comparative time series analysis. TiVEx achieves scalability through three complementary strategies: TiVEx-sharing exploits computational reuse across overlapping subsequence windows, eliminating redundant distance calculations; TiVEx-pruning employs distance-based upper bounds to eliminate unpromising candidates without exhaustive evaluation; and TiVEx-hybrid integrates both mechanisms to maximize efficiency gains. The key observation is that overlapping subsequences share a substantial computational structure, which can be systematically exploited while maintaining result optimality through provably correct pruning bounds. Extensive experiments on six diverse datasets demonstrate that TiVEx-hybrid achieves up to 84% reduction in distance calculations compared to exhaustive search while producing identical top- k results. Compared to state-of-the-art subsequence comparison methods, TiVEx-hybrid achieves $2.3\times$ improvement in computational efficiency. Our effectiveness analysis confirms that TiVEx achieves result quality within 5% of exhaustive search even when exploring only a subset of candidate positions, enabling scalable visual exploration without compromising insight quality.

Keywords: insight discovery; optimization; time series data; visual exploration

Academic Editors: María del Carmen Pegalajar Jiménez and Luis G. Baca Ruiz

Received: 9 November 2025

Revised: 28 January 2026

Accepted: 9 February 2026

Published: 16 February 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Visual exploration of time series data is essential for uncovering meaningful insights across domains such as finance, environmental monitoring, biomechanics, and healthcare [1–5]. However, exploring high-dimensional long time series remains challenging due to their inherent complexity and large volume. These challenges are amplified when analysts seek to identify localized insights hidden within long time series (e.g., [1–3,6]).

To discover localized insights in time series data, analysts often engage in visual exploration, selecting and plotting specific temporal windows (i.e., subsequences) for closer inspection. Throughout this process, analysts adjust parameters such as subsequence length and iteratively generate multiple visualizations that capture different segments across time series (e.g., [7–9]). Such manual exploration tends to be ad hoc, guided largely by the analyst's intuition and expertise. For instance, analyzing a single day of sensor data

recorded at minute-level resolution could involve examining thousands of subsequences, rendering manual exploration impractical.

To address these challenges, several automated approaches for visual time series exploration have been proposed (e.g., [10,11]), building on principles of interactive visual analytics that combine automated computation with human judgment to guide exploratory analysis [12,13]. These systems follow the visual analytics paradigm of generating candidate insights automatically while enabling analysts to interactively filter, rank, and examine results based on domain expertise [14]. However, existing systems primarily focus on insight discovery while overlooking the computational efficiency and scalability of the exploration process—an issue that becomes particularly severe for complex or long time series. This gap underscores the need for techniques that not only support visual insight discovery but also enable efficient and scalable exploration, a challenge that this work seeks to address.

In this work, we focus on subsequence comparison, a key analytical task in local time series exploration. Alongside correlation analysis, motif discovery, and anomaly detection [15–20], subsequence comparison enables the discovery of localized behaviors by measuring the similarities or differences between segments of time series. Our emphasis is on cross-dissimilarity comparison—comparing aligned subsequences from two time series and ranking them by their deviation values. This formulation is well suited for applications such as physiological monitoring (e.g., comparing signals from two sensors) and financial analysis (e.g., assessing synchronized stock-index movements) [21,22], where the alignment between time series must be preserved to capture meaningful, time-locked differences.

To illustrate the practical significance of automated subsequence comparison, consider the early detection of Parkinson’s disease using wearable motion sensors. In healthy individuals, sensors attached to the left and right ankles typically produce symmetric walking patterns. Subtle asymmetries—such as a reduced swing amplitude or a slight drag in one foot—may, however, appear months before clinically noticeable symptoms [23]. By automatically discovering and ranking aligned subsequences by their dissimilarity, analysts can pinpoint gait segments exhibiting the greatest asymmetry, enabling clinicians to focus on the most significant intervals instead of manually reviewing lengthy motion recordings.

It is important to distinguish this work from time series search methods. Search methods typically operate in a query-driven setting, where a known subsequence is provided and the goal is to locate similar or dissimilar matches, such as identifying motifs (repeated subsequences) or discords (anomalous subsequences) [15,24]. In contrast, time series data exploration does not rely on an input query. Instead, the goal is to uncover unknown and potentially interesting insights directly from the data itself. Rather than specifying what to look for, the analyst interacts with a system that systematically surfaces candidate insights for examination. Within this broader exploration context, our work focuses on the comparative analysis across two time series. Specifically, we aim to discover aligned subsequence pairs that exhibit high cross-dissimilarity—segments where the two series behave most differently and thus reveal the most significant variations or anomalies.

Existing approaches for time series exploration—particularly those addressing cross-dissimilarity analysis—often rely on threshold-based search techniques to identify (dis)similar subsequence pairs across two time series (e.g., [16–18,25–27]). Such approaches require analysts to manually specify a similarity or dissimilarity threshold, a process that is often unintuitive, difficult to tune, and highly dependent on dataset characteristics.

To address the limitations of threshold-based approaches, this work adopts a ranking-based exploration formulation, in which aligned subsequence pairs are ordered by their dissimilarity values, and the top- k results are selected for visualization. This formulation

removes the need for manually defining a distance threshold and provides a more intuitive control mechanism, allowing analysts to specify how many insights they wish to examine.

A straightforward way to realize this ranking-based exploration approach is to exhaustively compare all possible aligned subsequence pairs for a given subsequence length and then select the top- k most dissimilar pairs. However, this process is computationally prohibitive: the number of required comparisons increases quadratically with the length of the time series. For instance, a series of 30,000 points with a subsequence length of 50 would entail more than 1.4 million distance calculations, making interactive exploration infeasible for large datasets.

To address the computational cost of exhaustive comparison, we previously proposed TiVEx (Time Series Visual Exploration) [19], which identifies the top- k dissimilar subsequence pairs by exploiting shared computations among overlapping subsequence windows. This approach adapts the well-established concept of partial aggregation from data stream processing [28]—where intermediate results are reused across overlapping windows—to Euclidean distance computation for aligned subsequence comparison. We refer to this approach as TiVEx-sharing throughout this paper, and it serves as the baseline from which our novel contributions extend. While TiVEx-sharing substantially reduced redundancy between consecutive subsequences, it addressed only this single optimization dimension and overlooked optimization opportunities within individual subsequences (i.e., non-overlapping regions).

Hence, this paper introduces two novel techniques that extend beyond TiVEx-sharing. First, TiVEx-pruning (Section 6.1) introduces distance-based upper bounds that enable early termination when candidates provably cannot enter the top- k set, eliminating unnecessary point-pair computations within each subsequence. Second, TiVEx-hybrid (Section 6.2) integrates both sharing and pruning mechanisms, achieving maximum computational efficiency by reusing overlapping calculations across consecutive subsequences while simultaneously pruning unpromising candidates within non-overlapping regions. Importantly, both techniques maintain exact result optimality, producing identical top- k rankings to exhaustive search—efficiency gains are achieved solely by avoiding unnecessary computations, not by sacrificing accuracy.

Our TiVEx approach builds on the well-established principle of computation reuse in time series analysis. This principle underlies many efficient methods, including Matrix Profile methods [20,24,29–31], which achieve efficiency by reusing intermediate calculations when comparing subsequences across all temporal offsets, comparing every position in one series against every position in another. However, this generality comes at a cost: their sequential computation evaluates all positions before reaching any specific aligned pair. TiVEx avoids this overhead by leveraging the alignment constraint inherent to our problem. By directly targeting only temporally synchronized subsequences, we substantially reduce the search space: TiVEx-sharing reuses calculations between consecutive aligned pairs, while TiVEx-pruning and TiVEx-hybrid further exploit this constrained setting through the early termination of unpromising candidates.

Motivating Example. To highlight the value of the TiVEx approach, consider the example shown in Figure 1 applied to the Google Mobility dataset. This dataset, available at [32], has been widely used to analyze changes in human mobility patterns. Particularly, Google's data records changes in mobility across six different categories of places (e.g., residential, workplace, parks, etc.). Among the six categories, our focus is on workplace, which represents mobility related to work, and residential, which reflects time spent at home.

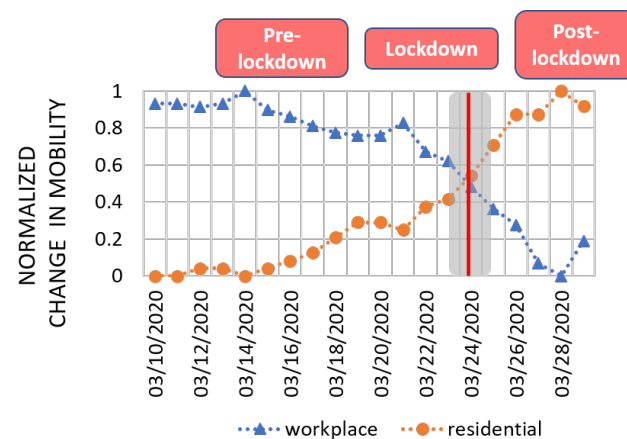


Figure 1. Workplace and residential mobility insights from the Google Mobility dataset before and after the COVID-19 lockdown, highlighting behavioral shifts.

Figure 1 illustrates that, in the UAE, the pre- and post-lockdown periods in March 2020 exhibited the most significant dissimilarities between these two categories. The figure demonstrates precisely the type of high-utility visualization that TiVEx targets: aligned subsequence pairs where the distance (e.g., Euclidean distance) between corresponding data points reaches its maximum. Prior to the lockdown, workplace mobility exceeded residential mobility, reflecting a normal routine in which individuals primarily commuted to work. However, after the lockdown, a clear shift occurred, with residential mobility increasing significantly while workplace mobility declined. This shift reveals the pandemic's dramatic impact on daily routines and societal behavior. This visualization is expected to be among the top- k discovered visualizations, capturing the highest dissimilarities between residential and workplace mobility. Manually identifying such insights across all possible aligned subsequence pairs would be computationally prohibitive, especially with large dataset sizes. This demonstrates the practical value of TiVEx's automated approach.

Hence, the contributions of our work are as follows:

- We introduce TiVEx, an optimization-based technique that significantly reduces computational cost by exploiting shared calculations between overlapping subsequences and incorporating early pruning strategies to avoid unnecessary computations.
- Our experiments demonstrate that TiVEx achieves speedups of $38.5\times$ over exhaustive search on real-world datasets while maintaining high-quality results, making interactive exploration feasible for time series data.
- We evaluate our TiVEx approach across six time series datasets, demonstrating its efficiency and effectiveness in discovering meaningful insights in various applications, such as gait asymmetry detection.

This paper is structured as follows: Related works are introduced in Section 2. Section 3 outlines the visual data exploration process. Section 4 discusses challenges in selecting top- k visualizations for time series data. Section 5 presents baseline solutions for our problem. Section 6 introduces our method for optimizing the exploration of insightful visualizations. Section 7 details the experimental testbed. Section 8 presents the performance analysis on real-world and synthetic datasets. This paper concludes with future directions in Section 9.

2. Related Works

This section presents the process of visual data exploration for time series data, various techniques for insight discovery, and optimization methods used to accelerate the exploration process. We also highlight how our work differs from existing studies.

Visual Insight Discovery Process. Visual data exploration is a critical aspect of data analysis, particularly when dealing with complex datasets, such as time series data. It involves utilizing graphical and visual representations to make data more understandable and to discover insights. In recent years, various visual analytic tools such as Tableau, Power BI, and Qlik have gained significant popularity due to their user-friendly interfaces and their ability to help users understand and explore data [7–9]. Despite the power of these tools, the insight exploration process heavily relies on the user's or analyst's knowledge of the data (i.e., user-driven), domain expertise, and familiarity with visualization tools and statistical analysis. Even domain experts can find manual insight discovery overwhelming. This is where automated visual exploration systems can play a vital role.

Such systems aid users in obtaining a more thorough and meaningful exploration of data automatically, resulting in valuable insights for decision-making and problem-solving (e.g., [10,11]). This can be done by swiftly traversing the space of visualization and exploring those that are deemed interesting. These systems support faster data exploration by reducing reliance on manual exploration. However, as the size and complexity of datasets grow, such systems face significant scalability challenges due to the quadratic nature of pairwise comparisons and the lack of efficient pruning strategies.

A key challenge in automated insight discovery is defining what makes a visualization or insight interesting. This can vary depending on the dataset, the analysts' objectives, and the context of exploration. Prior works have proposed various utility-driven methods, including deviation-based and similarity-based methods that rely on statistical distances, user-action-based methods that infer user intent from actions or historical data, and perception-based methods that learn human visual perception patterns [33–37]. The choice of metric is thus influenced by user interests and the specific domain in which the visual exploration process is applied. Among these, deviation-based metrics, such as Euclidean distance (ED) and Dynamic Time Warping (DTW), have proven particularly effective in identifying discord patterns (i.e., dissimilar segments) within time series data [20,24,30,38,39].

In [10], the researchers proposed a visual insight exploration system that helps to understand, explore, and visually identify relevant insights in any arbitrary data, including time series data. This system generates the top- k visual insights for various insight types, such as time series outliers. These visual insights are presented in ranked rows, allowing users to quickly identify the most relevant insights. Moreover, each insight type, represented by a row of visual insights, is ranked to identify the important and relevant insights in the data. Similarly, Visail [11], a reinforcement learning-based system, automatically explores eight different types of insights from time series data, including similarity and correlation. While these systems effectively generate insightful visualizations [10,11], our focus is on optimizing the computational process within the visual data exploration process, ensuring scalability while discovering visualizations with interesting insights.

Time Series Insight Exploration Techniques. Insight exploration in time series data encompasses various analytical tasks, including correlation analysis, motif discovery, anomaly detection, and subsequence comparison [18,20,24,27,29,30]. Our work focuses on subsequence comparison, which identifies (dis)similar segments across two time series. A common approach to this task is the time series join, which finds all pairs of subsequences that satisfy a given similarity criterion. For example, ARES (Adaptive Radius-Based Search) [25] uses Euclidean distance, while others (e.g., [16]) apply Dynamic Time Warping to identify matching subsequences. However, these approaches rely on user-defined thresholds (e.g., similarity > 0.8 or distance < 0.1), returning only pairs that meet the specified criterion. This threshold dependency can limit their ability to discover unexpected insights outside the predefined bounds.

In contrast, our methods eliminate thresholds by directly discovering the top- k most dissimilar aligned subsequence pairs. To contextualize our contribution, we compare against STOMP (Scalable Time Series Ordered-Search Matrix Profile) [31], a state-of-the-art Matrix Profile method that also avoids threshold constraints. We select STOMP as our baseline because it operates on pairs of time series to identify the most similar or dissimilar subsequences—the task most closely related to our cross-dissimilarity exploration. However, STOMP compares every position in one series against every position in the other, whereas our formulation targets only aligned positions. This fundamental difference in problem scope leads to different optimization strategies.

Specifically, STOMP computes distances between all pairs of subsequences to produce a complete distance matrix, from which the nearest (for motifs) or farthest (for discords) neighbors are identified at each position. This exhaustive computation prevents skipping any distance calculation, as every entry is needed for the final result. In contrast, our top- k ranking formulation enables early termination: once an upper bound indicates that a subsequence pair cannot enter the top- k set, we can safely skip the remaining calculations for that pair. Additionally, our alignment constraint substantially reduces the search space by comparing only temporally synchronized subsequences rather than all possible offset combinations.

Building on these observations, this paper extends our previous work, TiVEx-sharing [19], which introduced computation reuse for aligned subsequence comparison. While TiVEx-sharing reduced redundant calculations between consecutive aligned pairs, it did not exploit early termination. We propose TiVEx-hybrid, which integrates both computation sharing and upper-bound pruning, achieving maximum efficiency for discovering the top- k most dissimilar aligned subsequence pairs.

Optimizing the Insight Exploration Process. Time series data is inherently high-dimensional and complex, posing significant challenges for efficient processing. Over the years, various techniques have been developed to mitigate these challenges, with indexing and dimensionality-reduction techniques emerging as prominent categories. Indexing plays a crucial role in improving the efficiency of similarity search in large-scale time series data. Techniques such as Indexable iSAX [40] enable fast retrieval by structuring the data in a way that minimizes unnecessary distance computations. However, indexing is unnecessary when analyzing cross-dissimilarity between aligned pairs of subsequences from two time series. Since the locations of these subsequences are predetermined and accessed only once, comparisons are performed directly for each pair, eliminating the need for an index structure to speedup retrieval.

On the other hand, dimensionality-reduction techniques aim to transform high-dimensional time series data into a more compact representation while preserving key characteristics, enabling easier exploration and visualization. Examples of dimensionality-reduction methods include Principal Component Analysis (PCA) [41] and Singular Value Decomposition (SVD) [42]. While effective in reducing complexity, these methods come with significant preprocessing overhead, including parameter tuning, training, and computationally expensive transformations. This preprocessing overhead is particularly burdensome for the time series exploration process, where computational efficiency and rapid interaction with the data are critical.

Beyond indexing and dimensionality reduction, several optimization techniques have been proposed to improve the efficiency of (dis)similarity search in large-scale time series data. Lower-bounding techniques, such as envelope-based bounds [3,43], help prune unnecessary distance computations early by providing tight bounds in the search process. These techniques are particularly effective when applied to Dynamic Time Warping, significantly reducing the search space. However, for Euclidean distance, which lacks warping

flexibility, these bounds are often too loose to yield efficiency gains. Early abandoning techniques [3] have also proven effective for similarity search by terminating computations when the bounds indicate that no improvement is possible. However, these techniques have primarily focused on similarity search rather than dissimilarity ranking, requiring adaptation for top- k exploration scenarios.

Approximate nearest neighbor (ANN) methods, such as LSH [44] and KD-trees [45], accelerate similarity search by enabling fast, approximate retrieval within acceptable error thresholds. These methods focus on approximate nearest neighbors rather than ranked dissimilarity discovery and often require dataset-specific parameter tuning. To further improve scalability, parallel computing frameworks, including MapReduce and GPU acceleration, are employed to distribute computations across multiple cores or nodes. These frameworks significantly reduce execution time, especially for large dataset sizes. The integration of multiple optimization strategies, such as combining pruning with shared computation and parallel processing, can yield robust, scalable solutions for high-performance visual exploration. Our method takes a step in this direction by efficiently combining pruning and redundancy elimination to optimize top- k insight discovery in time series.

3. Preliminaries

This section provides the preliminary foundation for our work. Particularly, we formally define key concepts related to time series and subsequences (Section 3.1), followed by a description of the visual exploration process, with a specific focus on efficient subsequence comparison for insight discovery (Section 3.2).

3.1. Time Series Data

Definition 1 (Time series T). A time series T is an ordered sequence of observations over time, formally written as $T = \{x_1, x_2, \dots, x_L\}$, where L is its length.

Since the focus of this work is on the comparative exploration analysis of two distinct time series, a formal definition of an explored time series pair is formalized as

Definition 2 (Time Series Pair (T, T')). A pair of time series (T, T') consists of two sequences, T and T' , both of length L , representing data collected over the same temporal duration.

A subsequence $s_{b,e}$ in a time series T represents the segment between positions b and e , where $1 \leq b \leq e \leq L$. Hence, the subsequence length, denoted by R , is calculated as $R = e - b + 1$.

When comparing the two time series T and T' , an aligned subsequence pair refers to a pair of subsequences, $s_{b,e}$ from T and $s'_{b,e}$ from T' , that correspond to the same temporal interval (from b to e). Accordingly, each point in $s_{b,e}$ aligns with the point in $s'_{b,e}$ collected at the same timestamp.

Definition 3 (Subsequence View). A subsequence view, denoted as $v_{b,e}$, refers to the visualization of an aligned pair of subsequences ($s_{b,e}$ and $s'_{b,e}$) extracted from the time series T and T' , respectively. For brevity, this paper often refers to “a subsequence view” simply as “a view”.

For instance, in Figure 2, the subsequence view $v_{1,8}$ visualizes the aligned pair $(s_{1,8}, s'_{1,8})$ extracted from time series T (orange) and T' (blue), covering positions 1 to 8 with a subsequence length $R = 8$. All symbols used in this paper are summarized in Table 1.

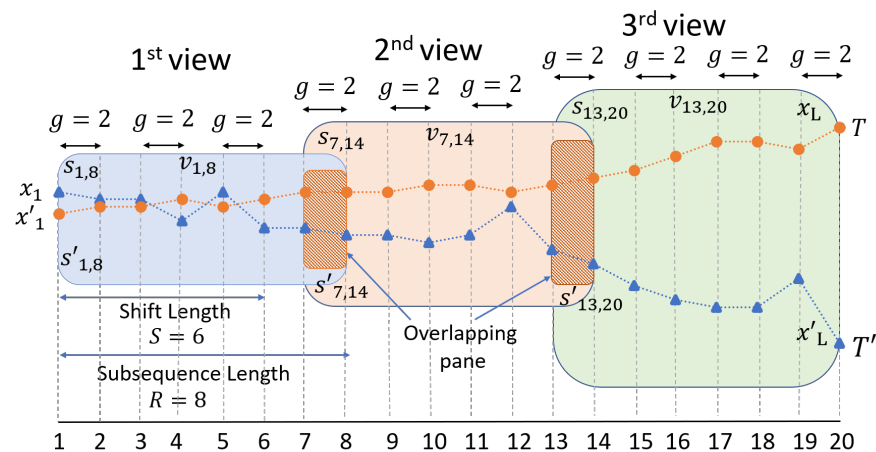


Figure 2. An illustration of sliding-window exploration over two time series T and T' of length $L = 20$, using subsequence length $R = 8$ and shift distance $S = 6$.

Table 1. Summary of symbols.

Symbol	Description
Time Series Characteristics	
L	Length of time series
R	Length of subsequence
S	Shift length between consecutive subsequences
n	Number of time series
T	A particular time series
x_i	Data point in T , $1 \leq i \leq L$
X_i	Normalized data point
Subsequences and Views	
$s_{b,e}$	Subsequence in T from position b to e
$v_{b,e}$	A particular view
$D(v_{b,e})$	Deviation of a view
$D_{\max}(v_{b,e})$	Maximum deviation of a view
v	All possible views
V	Number of views

3.2. Visual Data Exploration Process

With the formal definitions from Section 3.1 established, we now detail the time series visual data exploration process adopted in this work. Time series exploration typically operates at two levels: the global level, which examines the entire series to identify overall statistics or dominant trends, and the local level, which analyzes shorter subsequences to detect fine-grained behaviors or localized events. Both levels provide complementary insights—global exploration establishes the broader context, while local exploration uncovers specific events obscured when examining the entire time series.

This work focuses on local-level exploration because many critical insights of interest manifest as localized behaviors within subsequences rather than in aggregate statistics [18,20]. For instance, in financial markets, sudden price movements span only brief periods; in healthcare monitoring, irregular heartbeats occur as isolated events; and in sensor networks, equipment failures produce localized anomalies. These insights are often hidden when analyzing the full time series globally. Therefore, our process is centered on the efficient visual exploration of subsequences, enabling analysts to discover and investigate these local insights while maintaining awareness of the global context.

In this work, we focus on subsequence comparison, a key task in local time series exploration alongside correlation analysis, motif discovery, and anomaly detection [18,20,24,26,27,29–31]. Subsequence comparison is valuable in applications such as physiological monitoring and financial analysis [4,21], where time series are naturally aligned, and this alignment must be preserved to capture meaningful, time-locked differences.

Given two time series, T and T' , each with L data points, the goal is to discover aligned subsequence pairs of length R that reveal interesting insights. The subsequence length R is predefined based on domain knowledge (e.g., gait cycles or trading windows). This work focuses on identifying dissimilar subsequences, which highlight divergent behaviors, anomalies, or critical differences between time series.

To ensure that the resulting dissimilarity measure is meaningful and not dominated by differences in absolute magnitude, we must standardize the data. Therefore, to ensure consistent scaling across all views, we apply min–max normalization to both time series, T and T' , before the comparison process. Particularly, for T , each data point x_i is normalized to X_i using the formula $X_i = \frac{x_i - \min}{\max - \min}$, where $\min = \min_{i=1}^L(x_i)$ and $\max = \max_{i=1}^L(x_i)$ are the global minimum and maximum for T . The global normalization is also applied to T' , ensuring all values for both series lie in the range $[0, 1]$. Recall from Section 3.1 that the aligned subsequence pair $(s_{b,e}, s'_{b,e})$ constitutes the view $v_{b,e}$ that captures the synchronized behavior over the interval from b to e . Hence, the Euclidean distance or deviation $D(v_{b,e})$ for the view $v_{b,e}$ is calculated based on these normalized subsequences using Equation (1):

$$D(v_{b,e}) = \sqrt{\sum_{i=b}^e (X_i - X'_i)^2} \quad (1)$$

where X_i and X'_i are the normalized data points, and the positions b and e identify the start and end of the view $v_{b,e}$.

By normalizing each time series independently using its global min–max range, we remove differences in absolute magnitude between T and T' . This ensures that the dissimilarity measure compares the shape of the two series rather than being dominated by which series has larger absolute values. This is particularly useful for discovering different behavioral shifts across temporal periods, regions, or categories (e.g., Figure 1).

Identifying dissimilar subsequence pairs through exhaustive evaluation requires $O((L - R + 1) \times R)$ distance calculations, which becomes a significant computational bottleneck for long time series. For example, a time series with $L = 30,000$ and $R = 50$ requires over one million distance operations. Without optimization, this computational cost limits scalability to real-world, large-scale datasets. To address this challenge, this work introduces the TiVEx family of techniques (detailed in Section 6). TiVEx achieves computational efficiency through strategic computation sharing across overlapping views and early pruning of candidates that cannot qualify for top- k ranking. These optimizations reduce the cost of large-scale subsequence comparison, enabling timely insight discovery in practice.

4. Problem Definition

Having established the visual exploration process and its computational challenges in Section 3, we now formalize the problem addressed in this work. Our primary goal is to discover the top- k insightful visualizations (or views), where each view, as defined earlier, represents an aligned pair of subsequences from two time series. The process of finding these views is guided by a distance-based comparison function that quantifies the similarity

or dissimilarity between subsequence pairs, enabling the systematic identification of the most meaningful insights.

As discussed in Section 3, exhaustively evaluating all aligned subsequence pairs becomes computationally prohibitive for long time series. Even with a predefined subsequence length R , a distinct subsequence can start at every timestamp in the series, resulting in $O(L - R + 1)$ candidate pairs—an enormous search space for long sequences. To address this challenge, we adopt a sliding-window approach that balances exploration coverage with computational efficiency. The sliding-window formulation, widely adopted in data stream processing [46,47], provides a principled mechanism to control the trade-off between exploration granularity and computational cost through a shift-length parameter S .

The shift length S determines the stride between consecutive subsequences: when $S = 1$, the process enumerates all possible subsequences, achieving exhaustive exploration; when $S > 1$, the exploration becomes sparser, reducing both the number of candidate views and the computational cost. For meaningful overlap-based optimizations, we require $S < R$. When $S \geq R$, subsequences become non-overlapping, eliminating opportunities for computation sharing (discussed in Section 5.2).

Given a time series T of length L , subsequence length R , and shift length S , the sliding window generates a sequence of overlapping subsequences, where each new subsequence begins S time units after the previous one, as later illustrated in Figure 2. The total number of subsequences produced is $V = \frac{L-R}{S} + 1$. Each subsequence is formally expressed as s_{b_y, e_y} , where the start position $b_y = (y - 1)S + 1$ and end position $e_y = (y - 1)S + R$, for $1 \leq y \leq V$. For a pair of time series (T, T') , each view v_{b_y, e_y} corresponds to an aligned subsequence pair $(s_{b_y, e_y}, s'_{b_y, e_y})$, covering the same temporal interval. For each candidate view (e.g., $v_{1,8}$ or $v_{7,14}$, as later illustrated in Figure 2), we compute the distance metric defined in Equation (1) to quantify the dissimilarity between the aligned subsequences.

This work focuses specifically on views exhibiting high dissimilarity, as such deviations often reveal the most informative or unexpected temporal behaviors. Dissimilar regions indicate the periods where the two time series diverge significantly—precisely the insights most valuable for comparative analysis and understanding temporal variations. Accordingly, views are ranked by their distance values in descending order, with the top- k views representing the most dissimilar—and thus, the most insightful—subsequence pairs discovered during exploration. This ranking provides analysts with a prioritized list of visualizations that highlight the most distinctive differences between the time series.

In evaluating any approach to this problem, two main criteria are considered:

1. Efficiency: The ability to explore all possible views with minimal computational cost, enabling timely exploration of large time series (see Section 6).
2. Effectiveness: The ability to discover the top- k insightful views whose distance values closely match those obtained by the exhaustive search, ensuring that computational gains do not come at the expense of insight quality (see Section 7).

Based on our previous discussion, the problem addressed in this paper is defined as follows:

Definition 4 (Top- k Dissimilar View Discovery). *Given two time series (T, T') of length L , a subsequence length R , a shift length S (where $S < R$), and a positive integer k , the goal is to efficiently discover the top- k insightful views, where each view represents an aligned pair of subsequences of length R defined by the sliding-window parameter S . These views are ranked by their distance values, with larger distances indicating higher dissimilarity and, consequently, greater potential for analytical insight.*

This problem definition guides the design of the TiVEx family of techniques, which achieves efficient top- k discovery through strategic computation sharing (TiVEx-sharing), distance-based pruning (TiVEx-pruning), and their integration (TiVEx-hybrid), as detailed in Section 6.

5. Baseline Solutions

Before presenting our proposed method, we describe the baseline approaches used for comparison. While various techniques exist for accelerating the time series search—including indexing and dimensionality reduction—these are not directly applicable to our exploration problem. Search methods retrieve subsequences similar to a given query, whereas exploration systematically examines and ranks all aligned subsequence pairs to uncover insightful relationships. Indexing structures offer little benefit since exploration requires exhaustive traversal rather than selective access [40], and dimensionality-reduction techniques (e.g., PCA, SVD) involve parameter tuning and costly transformations that are impractical for interactive exploration [41,42]. These methods are discussed in the Related Work Section (Section 2); here, we focus on the baselines most relevant to the exploration task.

We therefore consider two primary baselines that reflect distinct computational strategies: (1) Exhaustive search, which provides a complete computational reference, and (2) TiVEx-sharing, our prior computation-sharing technique [19], which serves as a strong performance benchmark for comparison. Moreover, our experiments (Section 7) include two additional baselines—(3) a Sampled Exhaustive Search variant, which applies exhaustive search to a downsampled dataset to examine the trade-off between computational efficiency and insight quality, and (4) an adapted STOMP (Scalable Time Series Ordered-Search Matrix Profile) baseline [31], a shape-based approach using z-normalized distances to identify the most dissimilar aligned subsequences. Because both of these additional baselines build upon the exhaustive search concept, their detailed discussion is deferred to Section 7, after the exhaustive method is fully described in this section. These baselines collectively provide the yardsticks against which the computational efficiency and insight quality of the proposed technique are compared (see Section 7).

5.1. Exhaustive Search

The exhaustive search method serves as the reference baseline for evaluating computational efficiency. It performs a complete enumeration of all possible aligned subsequence pairs between the two time series. For each resulting view, the corresponding distance value is computed, and the top- k views with the largest distances are selected. This method yields the same set of insightful views as our proposed approach but requires evaluating every possible pair, leading to an extremely high computational cost for long time series.

Example. Consider the example in Figure 2, showing two time series T and T' of length $L = 20$. Using a subsequence length $R = 8$ and a shift length $S = 6$ yields three possible views ($V = 3$). For instance, the aligned subsequences $s_{1,8}$ and $s'_{1,8}$ form the first view $v_{1,8}$. In this case, exhaustive search computes the distance value for each of the three views, as defined in Equation (1), before selecting the top- k views with the largest distances.

Cost Analysis. To assess the efficiency of the exhaustive search method, we compute its total cost in terms of the number of required operations. The overall cost corresponds to the total number of operations needed to calculate the Euclidean distance for all views, as expressed in Equation (2):

$$\text{Total Cost} = (2 \times R - 1) \times V \quad (2)$$

The cost for computing the Euclidean distance within a single subsequence pair consists of two components:

1. Distance operations: The operations required to compute each of the terms $(X_i - X'_i)^2$ in Equation (1), performed R times for a subsequence of length R points.
2. Aggregate operations: The operations required to compute the overall summation $\sum_{i=b}^{b+R-1} (X_i - X'_i)^2$, performed $(R - 1)$ times for a subsequence of length R , assuming the subsequence starts at position b .

Combining the two components yields a total of $R + (R - 1)$ operations per view. Accordingly, the overall cost for V views equals $(2R - 1) \times V$, as shown in Equation (2). Notice that, since the Euclidean distance is monotonic, the square root can be omitted without affecting the ranking of views [3]. Putting it together, and returning to the example above, with $R = 8$ and $V = 3$, the total cost becomes $(2 \times 8 - 1) \times 3 = 45$ operations.

In the exhaustive search, the distance for each view is computed independently from scratch. To improve the efficiency of this process, the next section introduces an optimization strategy that significantly reduces redundant computations (see Section 6.2).

5.2. TiVEx-Sharing

Among the baselines, we include our previously proposed technique [19], which addresses the top- k dissimilar view discovery problem through computation sharing. This technique, which we refer to as TiVEx-sharing in this work, exploits overlapping subsequences between consecutive views to reuse pairwise distance calculations, substantially reducing computational redundancy. While TiVEx-sharing demonstrated the advantages of computation sharing between consecutive views, it relied exclusively on this mechanism for optimization. Here, we summarize its main principles for completeness, as the subsequent sections introduce additional techniques that build on and extend this foundation.

Computation Sharing. TiVEx-sharing reduces the computational cost by reusing intermediate results across overlapping subsequences, thereby avoiding redundant distance calculations between consecutive views. Each subsequence of length R is divided into equal-sized segments, or panes, whose size g is determined by the greatest common divisor of R and S ($g = \gcd(R, S)$). This alignment ensures that panes overlap consistently across successive views, enabling the direct reuse of their precomputed distances. This pane-based decomposition, widely adopted in streaming-query aggregation [28], forms the core of TiVEx-sharing's efficiency.

Figure 2 illustrates this process, where subsequences of length $R = 8$ and shift $S = 6$ yield a pane size $g = 2$ (the shaded areas indicate reused computations). A detailed numerical example is provided in Appendix A.

Partial Aggregation. Building on this pane-based structure, TiVEx-sharing performs partial aggregation within and across panes to further reduce redundant operations. Computation-sharing techniques commonly rely on partial aggregation, where intermediate results of aggregate functions (e.g., $\text{AVG}()$, $\text{SUM}()$, $\text{MIN}()$, $\text{MAX}()$) are reused across overlapping data segments [28,48]. TiVEx-sharing applies this principle to Euclidean distance computation by dividing each subsequence of length R into $\frac{R}{g}$ equal-sized panes, each treated as a partial aggregate. The distance for a view $v_{b,c}$ is then defined, as in Equation (3), where the inner summation computes the distance for all points within a pane. The outer summation combines these partial results across all panes, with c denoting the pane index.

$$D(v_{b,c}) = \sqrt{\sum_{c=1}^{R/g} \left(\sum_{i=b+g(c-1)}^{b+c \times g-1} (X_i - X'_i)^2 \right)} \quad (3)$$

Cost Analysis. We distinguish between two cases when analyzing the computational cost of TiVEx-sharing: the first view, where no reuse occurs, and the subsequent views, where overlapping computations are shared. For the first view, the cost is obtained by summing three components: (1) R distance operations for all points; (2) $(g - 1) \times \frac{R}{g}$ aggregation operations within panes; and (3) $\frac{R}{g} - 1$ final aggregation operations across panes. This yields a first-view cost of $(2R - 1)$ operations. Starting from the second view, TiVEx-sharing reuses overlapping aggregates from the preceding view [28] and recomputes only the new S points, combining them with $\frac{R-S}{g}$ overlapping partial aggregates. The total cost across V views is therefore

$$\text{Total Cost} = (2 \times R - 1) + (2 \times S - 1 + \frac{R - S}{g}) \times (V - 1) \quad (4)$$

In Equation (4), the first term represents the initial view, while the second term corresponds to the per-view cost for subsequent views, where partial aggregation is applied. Because $S \leq R$, as S increases, increasing S reduces the overall computational cost relative to exhaustive search. A detailed numerical example of these calculations is provided in Appendix A.

Summary. TiVEx-sharing effectively reduces redundant distance computations by reusing overlapping aggregates between consecutive views, demonstrating the benefit of computation sharing in the top- k dissimilar view discovery task. However, its optimization focuses solely on reuse between adjacent views and does not eliminate redundant computations within each view. The next section introduces our proposed technique, which extends this foundation with additional optimization strategies to enhance computational efficiency.

6. Proposed Approach

While TiVEx-sharing effectively reduces redundant computations by reusing overlapping aggregates between consecutive views, it relies solely on computation sharing and overlooks unnecessary distance calculations that arise within each view. To overcome this limitation, we introduce two additional techniques: TiVEx-pruning (Section 6.1) applies distance-based upper bounds to enable early termination when candidates cannot enter the top- k , and TiVEx-hybrid (Section 6.2) integrates both pruning and sharing to maximize efficiency gains. Together, these three techniques—TiVEx-sharing, TiVEx-pruning, and TiVEx-hybrid—form a complete family of algorithms that address different aspects of computational efficiency in top- k dissimilar view discovery.

6.1. Pruning of Computation

The pruning strategy in TiVEx aims to eliminate unnecessary distance calculations by determining, during computation, when a candidate view can no longer qualify for the top- k dissimilar set. Instead of evaluating every point pair within a view, the method incrementally estimates the maximum possible deviations that the remaining points could contribute and terminates the computation once this bound falls below the current top- k threshold (Dev_{ref}). This selective evaluation ensures substantial computational savings without compromising the correctness of the final ranking.

Early-Abandoning Principle. The early abandoning technique [3], extensively used in time series analysis tasks, allows for the premature termination of distance calculations when it becomes evident that a candidate cannot achieve a target similarity threshold. In a traditional similarity search, this technique stops distance computation as soon as the partial distance exceeds the current best match, since continuing the calculation cannot improve the result.

However, our problem in TiVEx requires identifying the maximum dissimilarity for top- k ranking, which inverts the traditional early abandoning logic. Instead of abandoning when distances become too large, we abandon when the maximum possible distance (i.e., upper bound) becomes too small to enter the top- k ranking. This fundamental adaptation transforms early abandoning from a similarity optimization into a dissimilarity ranking optimization.

Incremental Processing Strategy. Rather than computing all pairwise distances for all point pairs at once, our approach progressively incorporates point pairs into the dissimilarity calculation. We maintain an upper-bound estimate of the maximum possible deviation that a view can achieve, enabling early termination when this bound falls below our ranking threshold (Dev_{ref}). For any given view, we categorize point pairs into two types: known and unknown. Known pairs have already been processed and add their actual normalized squared deviations $(X_i - X'_i)^2$ to the cumulative sum. Unknown pairs have not yet been processed and are assumed to contribute the maximum possible deviation after normalization, $(1 - 0)^2 = 1$, as a worst-case estimate. The deviation upper bound is then the sum of the actual distances for known pairs and one for each unknown pair. The mathematical validity of this upper-bound calculation is established by the following lemma (proof in Appendix B):

Lemma 1 (Upper-Bound Validity). *Given normalized data points $X_i, X'_i \in [0, 1]$ for a view $v_{b,e}$, suppose point pairs at positions $b, \dots, j$ have been processed (where $b \leq j \leq e$), while positions $j + 1, \dots, e$ remain unprocessed. Then*

$$D_{\max}(v_{b,e}) = \sqrt{\sum_{i=b}^j (X_i - X'_i)^2 + (e - j)}$$

is a valid upper bound, satisfying $D(v_{b,e}) \leq D_{\max}(v_{b,e})$.

Remark 1. *This upper bound enables early abandoning by providing a conservative estimate of the maximum deviation a view can achieve. During the exploration process, when the upper bound of a candidate view falls below the minimum deviation in the current top- k set (i.e., threshold), we can safely prune the remaining unknown point pairs from positions $j + 1$ to e , as the view cannot enter the top- k ranking even under the scenario where all unknown pairs achieve the maximum distance.*

Example. Consider a view with $R = 4$ points. Assume that after processing the first 2 points, the squared distances are 0.0 and 0.01, respectively. The upper bound for the maximum possible deviation is then $D_{\max}(v_{b,e}) = \sqrt{0.0 + 0.01 + 2 \times 1} = \sqrt{2.01} \approx 1.418$, where the 2×1 term represents the maximum possible squared distance for the remaining 2 unprocessed points. Now assume the current ranking threshold is $Dev_{ref} = 1.5$ (the minimum deviation among the top- k views discovered so far). Since $1.418 < 1.5$, the remaining 2 points can be safely skipped without computing their distances. The view provably cannot enter the top- k set, even in the best case, where the remaining points achieve maximum dissimilarity.

This ensures a progressively tighter bound while maintaining computational efficiency. In the best case, when early abandoning is immediately triggered, the computational cost reduces from $O(R)$ to $O(1)$ per view. In practice, savings are proportional to the number of pruned point pairs.

Theorem 1 (TiVEx-Pruning Exactness). *TiVEx-pruning returns exactly the same top- k views as an exhaustive search.*

Proof. By Lemma 1, for any view $v_{b,e}$, we have $D(v_{b,e}) \leq D_{\max}(v_{b,e})$. When TiVEx-pruning detects that $D_{\max}(v_{b,e}) < Dev_{ref}$, the true deviation satisfies $D(v_{b,e}) \leq D_{\max}(v_{b,e}) < Dev_{ref}$. Therefore, the remaining point-pair computations within $v_{b,e}$ are safely skipped, and $v_{b,e}$ cannot enter the top- k ranking even under best-case conditions. Since TiVEx-pruning discards only views that the exhaustive search would also reject, the final top- k results are identical. $\square$

Pruning-Based TiVEx Approach. We refer to the pruning-based version of TiVEx as TiVEx-pruning. It combines the early-abandoning principle with incremental upper-bound estimation to avoid redundant distance calculations within each view. The algorithm operates sequentially across consecutive views, progressively refining deviation bounds and applying pruning at the point-pair level rather than after completing an entire view. This fine-grained control enables early termination at more precise stages of computation, resulting in substantial efficiency gains compared with traditional view-level pruning used in similarity search.

Figure 3 illustrates the pruning workflow for each candidate view $v_{b,e}$. The algorithm maintains an upper bound $D_{\max}(v_{b,e})$ (Lemma 1) that tightens as point pairs are incrementally processed. When the bound falls below Dev_{ref} , the remaining point pairs are safely pruned without affecting result optimality.

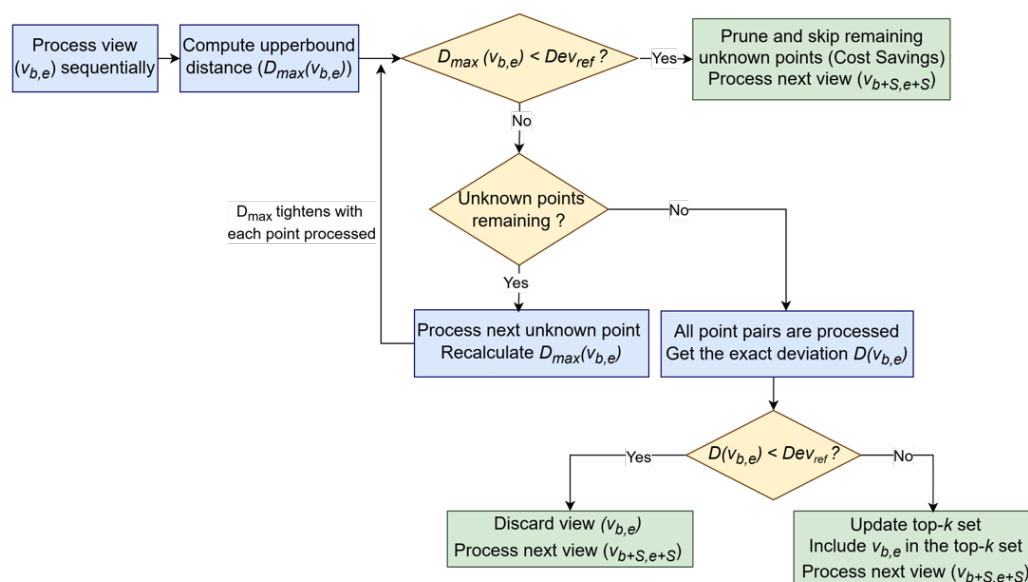


Figure 3. Workflow of the TiVEx-pruning technique for processing each candidate view.

Algorithm Overview. Our algorithm operates by maintaining a ranking threshold Dev_{ref} (the minimum deviation among current top- k views). For each candidate view, it incrementally calculates an upper bound on possible deviation using known point pairs (actual distances) and unknown pairs (maximum possible values). When this upper bound falls below Dev_{ref} , the remaining point pairs are pruned and the view is discarded, avoiding unnecessary computation. Views that pass the early abandoning condition are fully evaluated and integrated into the top- k ranking if their deviation exceeds the threshold.

Algorithm Details. The detailed steps of this technique are presented in Algorithm 1, which outlines the initialization of deviation calculations, iterative upper-bound refinement, and early-termination conditions for each view.

At the beginning, the deviation of the first view in the time series $D(v_{1,R})$, treated as a special case, is calculated by initially considering all points' normalized values (i.e., actual distance) (line 5 of Algorithm 1). This deviation is then stored in the reference deviation list, denoted as $DevList$. This list is initialized with a very small value (e.g., $-\infty$) and has a length of k to ensure that the first k views enter the list, establishing the initial ranking

baseline. Moreover, it is sorted in descending order, with the last value representing the minimum of the largest deviations (i.e., k^{th} largest deviation encountered so far), referred to as Dev_{ref} . As we progress to the following views, the deviation upper bound (i.e., maximum distance) is initially calculated by considering all point pairs as unknown.

Algorithm 1 TiVEx-pruning technique

```

1: Input: Time series  $T, T'$ , parameters  $R, S, k$ 
2: Output: Top- $k$  views with highest dissimilarity
3: Initialize a sorted list  $DevList$  of length  $k$  with  $-\infty$  values.
4: Set  $Dev_{ref} = DevList[-1]$  ▷ Minimum value in top- $k$  deviations.
5: Calculate  $D(v_{1,R})$  using Equation (1) for all  $R$  point pairs. ▷ First view: no overlap, compute all distances.
6: Store  $D(v_{1,R})$  in  $DevList$ .
7: for each view  $v$  do ▷ Iterate over views excluding the first one.
8:   while unknown point pairs exist do
9:     Calculate the deviation upper bound  $D_{\max}(v)$ . ▷ Valid by Lemma 1
10:    if  $Dev_{ref} > D_{\max}(v)$  then
11:      Prune unknown point pairs from deviation calculations.
12:      Continue to the next view.
13:    else
14:      Convert the next unknown point pair to known by using its actual normalized value.
15:      Recalculate  $D_{\max}(v)$  with updated known/unknown pairs.
16:      Continue the previous steps.
17:    end if
18:  end while
19:  Calculate the deviation  $D(v)$ .
20:  if  $Dev_{ref} > D(v)$  then
21:    Continue to the next view.
22:  else
23:    Replace  $Dev_{ref}$  with  $D(v)$ .
24:    Re-sort  $DevList$  in descending order.
25:    Update  $Dev_{ref} \leftarrow$  last element of  $DevList$ .
26:    Continue to the next view.
27:  end if
28: end for
29: return the  $k$  views with the largest deviations stored in  $DevList$ .
  
```

To refine the deviation upper bound, our algorithm incrementally incorporates point pairs in sequential order into the deviation calculation, marking them as known points. In this case, the refined upper bound is calculated based on the known and unknown point pairs within the view. After recalculating the upper-bound distance for a view, our algorithm compares this value to the last value in the sorted list $DevList$ (i.e., Dev_{ref}), which represents the ranking threshold. When the refined upper-bound distance is less than Dev_{ref} , this triggers the early abandoning mechanism, meaning that the view cannot be part of the top- k discovered views (lines (10–12) of Algorithm 1). Consequently, the unknown point pairs are pruned from further deviation calculations, and the algorithm proceeds to the consecutive view. This is where the computational cost savings are achieved through the adapted early abandoning principle and incremental processing strategy.

However, if the previous condition is not satisfied and there are still unknown point pairs within the view, each unknown point pair is incorporated incrementally into the deviation calculation using its actual normalized value (i.e., the exact distance). After each inclusion, we recalculate the deviation upper bound for the view and re-evaluate the early abandoning condition (lines (13–16) of Algorithm 1). This iterative process—incorporating unknown pairs,

updating the upper bound, and checking the abandoning condition—continues until all point pairs within the view are marked as known or pruned. The pruned pairs directly contribute to computational savings, thereby improving overall efficiency.

Upon reaching the stage where all point pairs become known (no early abandoning occurred), if the deviation of a view is less than Dev_{ref} , then the algorithm continues to the next view (lines (20–21) of Algorithm 1). This indicates that all data point pairs are incorporated into the deviation calculation and that no pruning occurs. However, if the condition remains unsatisfied, the deviation of the view replaces Dev_{ref} in the list $DevList$ and continues to the subsequent view (lines (22–26) of Algorithm 1). Notably, this deviation must be compared with the other deviations in the list $DevList$ to ensure that the list remains sorted.

6.2. Integrated Scheme

While TiVEx-sharing and TiVEx-pruning each provide substantial efficiency gains, neither alone fully addresses the computational challenges of large-scale time series exploration. TiVEx-sharing reduces inter-view redundancy but evaluates all points within each view, while TiVEx-pruning enables intra-view early termination but ignores cross-view overlaps. We introduce *TiVEx-hybrid*, which integrates both mechanisms to maximize efficiency by reusing pane-level aggregates across overlapping views (sharing) while eliminating unnecessary calculations within non-overlapping portions (pruning).

TiVEx-hybrid operates as follows: each subsequence is segmented into non-overlapping panes of size $g = \gcd(R, S)$. For overlapping panes between consecutive views, TiVEx-sharing reuses previously computed distances, eliminating redundant calculations. For the non-overlapping panes unique to each view, TiVEx-pruning applies incremental upper-bound refinement and early termination, pruning point pairs that cannot contribute to the top- k set. This two-level optimization strategy enables TiVEx-hybrid to systematically minimize computational cost while maintaining result correctness.

Hence, within TiVEx-hybrid, each view $v_{b,e}$ consists of two types of partial aggregates (or panes) based on the panes in the previous overlapping view $v_{b-S,e-S}$ as follows:

1. **Seen Panes:** The panes in $v_{b,e}$ have already contributed to the deviation calculation of the previous overlapping view $v_{b-S,e-S}$. These previously computed results are reused directly by leveraging our TiVEx-sharing, thereby reducing redundant calculations.
2. **Unseen Panes:** The panes in $v_{b,e}$ have not been involved in prior deviation calculations $v_{b-S,e-S}$. To optimize efficiency, TiVEx-pruning is applied to determine whether these panes can be pruned from further computations.

The TiVEx-hybrid technique follows a similar iterative process to TiVEx-pruning but operates at the pane level rather than at the point level. Unlike TiVEx-pruning, which processes individual point pairs, TiVEx-hybrid processes entire panes of size g , providing computational efficiency at the cost of a slightly coarser granularity.

Algorithm Overview. Our algorithm integrates both optimization strategies by operating at the pane level. For each view, it first reuses distances from overlapping (seen) panes computed in the previous view, leveraging TiVEx-sharing to eliminate redundancy. For unseen panes, it applies TiVEx-pruning logic by incrementally calculating upper bounds and applying early termination when the bound falls below the threshold Dev_{ref} . This dual-level optimization prunes entire panes rather than individual point pairs, providing greater computational savings by exploiting both inter-view and intra-view redundancies.

The detailed steps for applying the TiVEx-hybrid technique (presented in Algorithm 2) are as follows: after partitioning the subsequences, the exact deviation of the first view $D(v_{1,R})$ is calculated by initially considering the actual distances of all points in all panes using Equation (3) with $b = 1$ and $e = R$. This deviation is then stored in the reference deviation list $DevList$ (line 4 in Algorithm 2).

Algorithm 2 TiVEx-hybrid technique

```

1: Input:  $T, T'$ , parameters  $R, S, k$ 
2: Output: Top- $k$  views with highest dissimilarity
3: Initialize  $DevList$  (length  $k$ ),  $Dev_{ref} = DevList[-1]$ ,  $g = \gcd(R, S)$ 
4: Compute  $D(v_{1,R})$  using all panes of size  $g$  (using Equation (3)); Store in  $DevList$ 
5: for each view  $v_{b,e}$  do
6:   Reuse distances for seen panes from  $v_{b-S,e-S}$  ▷ Sharing
7:   while unseen panes exist do ▷ Pruning
8:     Calculate  $D_{\max}(v_{b,e})$  from seen and unseen panes (3 cases in Figure 4)
9:     if  $D_{\max}(v_{b,e}) < Dev_{ref}$  then
10:       Prune all unseen panes; break
11:     else
12:       Compute distance considering the next unseen pane; mark as seen
13:     end if
14:   end while
15:   if all panes processed then
16:     Calculate  $D(v_{b,e})$ ; Apply previous condition
17:     Update  $DevList$  and  $Dev_{ref}$  if needed
18:   end if
19: end for
20: return Top- $k$  views from  $DevList$ 

```

As we proceed to the subsequent views, each subsequence contains seen and unseen panes. Thus, the upper bound of deviation for each view, except the first view, is calculated by summing two components: (1) the pairwise distance between point pairs in seen panes, directly derived from the previous view, and (2) the maximum possible distance (i.e., $(1 - 0)^2$) for each point pair in unseen panes.

The calculation of the deviation upper bound for a particular view $v_{b,e}$ depends on the overlapping panes with the previous view $v_{b-S,e-S}$, leading to three distinct cases, as shown in Figure 4:

1. All seen panes: If all aggregates in a view $v_{b-S,e-S}$ are seen (i.e., from position $b - S$ to $e - S$), all overlapping aggregates from position b to $e - S$ are marked as seen panes in the subsequent view $v_{b,e}$, as shown in Figure 4a. In this case, the deviation upper bound $D_{\max}(v_{b,e})$ is calculated using Equation (5). The distance calculations for the first three panes in $v_{b,e}$ (i.e., the first term in Equation (5)) are directly derived from $v_{b-S,e-S}$.

$$D_{\max}(v_{b,e}) = \sqrt{\sum_{c=1}^{(R-S)/g} \mathbb{P}_c} + \sqrt{\sum_{c=1}^{S/g} \left(\sum_{i=b+g(c-1)+R-S}^{e+c \times g-S} (1-0)^2 \right)} \quad (5)$$

where $\mathbb{P}_c = \sum_{i=b+g(c-1)}^{b+c \times g-1} (X_i - X'_i)^2$.

2. All unseen panes: As illustrated in Figure 4b, if all aggregates in the overlapping region (i.e., from position b to $e - S$) are unseen in a view $v_{b-S,e-S}$, the deviation upper bound of $v_{b,e}$ is calculated as shown in Equation (6), where all panes are considered unseen.

$$D_{\max}(v_{b,e}) = \sqrt{\sum_{c=1}^{R/g} \left(\sum_{i=b+g(c-1)}^{b+c \times g-1} (1-0)^2 \right)} \quad (6)$$

3. Some seen/unseen aggregates: If some overlapping panes in view $v_{b-S,e-S}$ are seen while others are unseen, as depicted in Figure 4c, the deviation upper bound $D_{\max}(v_{b,e})$ is computed by separately summing the distances of seen aggregates (derived from $v_{b-S,e-S}$) and unseen aggregates as follows: $D_{\max}(v_{b,e}) = \sqrt{\mathbb{P}_1 + 3 \times 2 \times (1 - 0)^2}$. This example shows the specific case with one seen pane and three unseen panes; the general formula adapts based on the actual seen/unseen pane distribution.

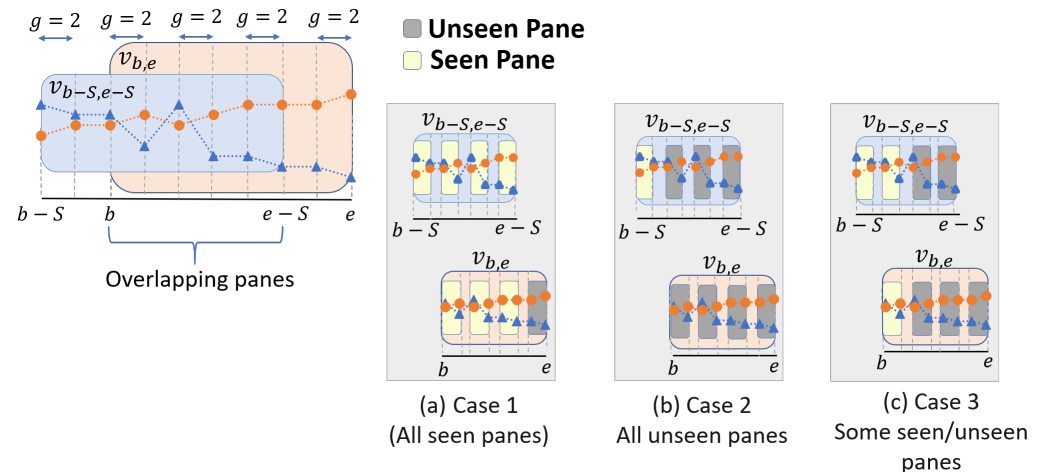


Figure 4. Three cases for calculating the deviation upper bound when applying TiVEx-hybrid to the views resulting from two time series T and T' with parameters $L = 10$, $R = 8$, and $S = 2$.

These three cases cover all possible configurations of pane overlap between consecutive views, ensuring that the upper bound of deviation can be computed efficiently under any conditions.

Theorem 2 (TiVEx-Hybrid Exactness). *TiVEx-hybrid returns exactly the same top- k views as the exhaustive search.*

Proof. Since each pane is an aggregate of exact point distances, the upper-bound computation at the pane level is mathematically equivalent to point-level computation in Theorem 1. Therefore, the pruning logic of Theorem 1 directly applies: views with $D_{\max}(v_{b,e}) < Dev_{ref}$ are safely discarded, and the final top- k results are identical to the exhaustive search.

□

This upper-bound distance for a view (i.e., $D_{\max}(v_{b,e})$) is then compared with the last value in the list $DevList$ (i.e., Dev_{ref}), using the same conditions and steps employed in the TiVEx-pruning technique. However, if $D_{\max}(v_{b,e}) \geq Dev_{ref}$, the number of point pairs incorporated in the deviation calculation is based on the pane size (line 12 in Algorithm 2). This process is repeated until the aggregates of the view are either marked as seen or the point pairs within the unseen panes are pruned (line 14 in Algorithm 2). Conversely, if $D_{\max}(v_{b,e}) < Dev_{ref}$, all point pairs in unseen aggregates are pruned from any further deviation calculations (i.e., cost savings); this view cannot be among the top- k discovered views (lines (9–10) in Algorithm 2). Note that abandoning occurs at the pane level, such that entire unseen panes are pruned rather than individual point pairs, making the process more efficient but less granular than TiVEx-pruning. Finally, the top- k views with the largest deviations are retained in the $DevList$ list.

This approach efficiently exploits the overlapping, seen aggregates between views while minimizing unnecessary operations in the remaining portions. TiVEx-hybrid achieves

$O(g)$ granularity for early abandoning (i.e., best case) by processing g point pairs per abandoning decision compared to $O(1)$ in TiVEx-pruning, while considering $O((R - S)/g)$ computation-sharing benefits from TiVEx-sharing in the best case, thereby providing greater computational savings.

7. Experimental Testbed

This section presents a comprehensive experimental evaluation assessing both efficiency (computational cost and runtime) and effectiveness (quality of the discovered insights). Our evaluation employs six diverse datasets, four baseline methods, including state-of-the-art approaches, and various parameter configurations.

7.1. Baseline Methods

We compare our proposed techniques against four carefully selected baselines (the first two are detailed in Section 5), each serving a specific evaluation purpose:

- Exhaustive search provides the baseline results by exploring all possible aligned subsequence pairs (or views). This establishes the computational upper bound for comparison (detailed in Section 5.1).
- TiVEx-sharing [19], our previously proposed method, leverages computation sharing among overlapping subsequences. This baseline, detailed in Section 5.2, enables the evaluation of the improvements achieved by the TiVEx-hybrid technique over our previous approach.
- Sampling Exhaustive Search represents a data reduction strategy: downsample the input time series at the rate r_s , then apply exhaustive search on the reduced data [49]. This baseline tests whether simple preprocessing can match algorithmic optimization, trading data quality for computational savings. We refer to this method as *sampling* in our experiments.
- STOMP [31] is a state-of-the-art method for time series analysis based on the Matrix Profile framework. Unlike our aligned pair formulation, STOMP computes distances between all pairs of fixed-length subsequences across two time series, comparing every subsequence position in the first series against every position in the second. STOMP achieves efficiency by exploiting redundancy when sliding across successive positions: as it compares one subsequence against all positions in the second series, it reuses overlapping computations from one comparison to the next. This optimization differs fundamentally from our TiVEx-sharing approach, which reuses computations along the aligned path between successive subsequence pairs. Whereas STOMP implements a general, multidirectional reuse mechanism designed for an exhaustive analysis of all pairs, TiVEx-sharing introduces a targeted reuse formulation specialized for efficient alignment-based comparison, prioritizing speed and interpretability over generality. Accordingly, to adapt STOMP to our setting, we extract only the main-diagonal entries from its complete Matrix Profile (aligned pairs) and select the top- k dissimilar ones. Importantly, STOMP itself remains unmodified; it computes the full Matrix Profile using standard incremental calculations with overlapping reuse. We then extract the diagonal entries, which naturally correspond to aligned subsequence pairs in our problem setting.

7.2. Datasets

In our experiments, we utilize six datasets: four real-world datasets and two synthetic datasets. These datasets were chosen to capture a diverse range of temporal insights and variations in dataset length. The real-world datasets provide naturally occurring trends, while the synthetic datasets offer controlled data for analyzing stochastic behaviors and

testing edge cases. This allows for a thorough assessment of our techniques. Below, we provide detailed descriptions of each one, along with their respective roles in our evaluation:

Real-World Datasets:

- **Google Mobility dataset:** Part of our experimentation relies on the Google Mobility dataset [32], which serves as the default dataset in our analysis. We selected a subset of this data with a length of $L = 600$, covering the period from 15 February 2020 to 6 October 2021. It encompasses six distinct categories of time series data (e.g., workplace, residential, parks, etc.) across various regions and countries. In our experiment, we explore pairs of time series data (specifically, workplace and residential data) from five different countries: the UAE, the US, Brazil, South Korea, and Australia. All results reported on this dataset are based on residential and workplace time series pairs and are averaged across the five countries to ensure robustness and generalizability.
- **Stock market dataset:** This dataset is obtained from Yahoo Finance [22], which provides historical time series for a wide range of financial instruments (e.g., stocks, commodities, and indices). For our experiments, we selected two representative commodity time series: gold and crude oil prices. The dataset spans from 1 January 2014 to 31 December 2021, and includes 2012 aligned daily time points. These commodities are chosen to represent real-world financial trends, providing a diverse context for evaluating the performance of our techniques.
- **WearGait-PD:** Designed to support gait analysis in individuals with Parkinson's disease (PD), this dataset contains rich multivariate time series data obtained from wearable inertial measurement units (IMUs) and smart insoles [21]. For our evaluation, we selected a walking trial from one PD patient who performed a free walk along predefined paths, including mid-walk turns and potential stair navigation to mimic real-world movement conditions. The trial spans a duration of 145.43 s, sampled at 100 Hz, yielding a total of 14,544 data points. The dataset is well suited for discovering gait asymmetry and supports the identification of aligned subsequences that exhibit strong dissimilarity between left and right limb movements. Our experiment specifically focuses on the linear acceleration data from the left and right ankles.
- **Seismic dataset:** We employ seismic data from the IRIS FDSN Global Seismic Network, specifically from station ANMO (Albuquerque, New Mexico) [50]. The recording, obtained on 15 January 2024, from 00:30:00 to 01:30:00 UTC, contains one hour of continuous ground motion data sampled at 40 Hz, yielding 144,000 samples. The dataset comprises three orthogonal acceleration components—vertical (Z), north–south (N), and east–west (E)—representing ground motion in different directions. For this paper, we focus on aligned subsequence pairs between the vertical (Z) and north–south (N) components. The large sample size and distinct component relationships make this dataset valuable for evaluating our top- k dissimilar pair detection across diverse temporal structures and component dissimilarity insights.

Synthetic Datasets:

- **Random walk dataset:** The random walk dataset consists of 4000 synthetically generated data points, where each step follows a Gaussian distribution with a mean of 0 and a standard deviation of 1. The values evolve cumulatively, ensuring that each point is influenced solely by the sum of previous random variations. This makes the dataset entirely stochastic and unbiased. By using this dataset, we aim to evaluate our techniques under scenarios with no inherent structure or predictable insights.
- **Sine–Cosine dataset:** This synthetic dataset was designed to control the position of the maximum deviation within the time series. By adjusting the phase shift between sine and cosine signals, three configurations were created in which the highest deviation occurs at the beginning, middle, or end of the 4000-point time series. This enables a

systematic evaluation of pruning performance across different deviation positions. Detailed scenarios are presented in Section 8.

7.3. Parameter Configuration

Table 2 summarizes the parameter settings used across all the experiments and serves as a reference point throughout the evaluation. Default parameters were selected based on preliminary experiments and domain knowledge. For example, subsequence length $R = 24$ represents approximately 4% of the default dataset length, providing meaningful insight detection. Similarly, the shift length $S = 6$ was chosen to strike a balance between overlap and separation, ensuring sufficient temporal distinction between successive views without incurring excessive computational costs.

Table 2. Experimental parameters with default values in bold.

Parameter	Values (Default in Bold)
Dataset	Google Mobility ($L = 600$), Stock Market ($L = 2012$), WearGait-PD ($L = 14,544$), Random Walk ($L = 4000$), Sine-Cosine ($L = 4000$), Seismic ($L = 144,000$)
Methods	Exhaustive Search, Sampling, TiVEx-Sharing, STOMP, TiVEx-Pruning, TiVEx-Hybrid
R	[5–60] (24)
S	1, 2, 3, 4, 6 , 9, 12 (STOMP: $S = 1$ only)
k	[1–7] (4)
r_s	0.2, 0.4, 0.5 , 0.6, 0.8

It is important to note that the dataset length is not a primary concern in our experiments. Our evaluation focuses on scalability: how the number of distance operations grows with dataset length and how efficiently our techniques reduce this computational cost. The diverse dataset lengths ($L \in \{600, 2012, 4000, 14,544, 144,000\}$) allow us to assess scalability across different problem sizes.

7.4. Performance Metrics

All methods are evaluated using two metrics that assess both computational efficiency and solution quality:

- (1) Efficiency: We quantify efficiency using three complementary measures: (i) Computational cost: The total number of distance operations required to process all discovered views, as detailed in Section 6. (ii) Execution time: Actual wall-clock runtime across different techniques and parameter configurations, validating theoretical complexity with empirical evidence. (iii) Normalized savings: The percentage of operations saved relative to the exhaustive search.
- (2) Effectiveness: We assess solution quality by directly measuring the deviation values of the discovered views. For any method and any shift length S , we compute the average deviation of the top- k discovered views as

$$\text{Average Deviation} = \frac{1}{k} \sum_{i=1}^k D(v_i) \quad (7)$$

where $D(v_i)$ denotes the deviation value of the i -th ranked view. Since we focus on dissimilarity-based exploration, higher average deviation values indicate that a method successfully identifies more distinctive views. This metric is applied to all methods regardless of the shift length S used.

The shift length S controls the exploration granularity and creates an efficiency–effectiveness trade-off. With $S = 1$, all consecutive positions are explored, providing

the finest granularity and typically yielding the highest average deviation (the most comprehensive exploration). As S increases, fewer positions are explored (every S -th position), reducing computational cost but potentially missing some highly dissimilar views located between sampled positions. Our evaluation systematically examines this trade-off by comparing average deviation values across different shift lengths for each method.

We use an exhaustive search with $S = 1$ as our primary comparison point because it provides the most comprehensive exploration, examining all possible consecutive positions. However, this does not imply it is the only “ground truth”; any method (including TiVEx-sharing, TiVEx-pruning, or TiVEx-hybrid) operating at $S = 1$ produces identical results to exhaustive search at $S = 1$, as they all evaluate the same positions. The key distinction is efficiency: our methods reduce the computational cost through sharing and pruning optimizations while producing identical rankings when applied to the same positions. For $S > 1$, we compare how different methods maintain solution quality (average deviation close to the $S = 1$ baseline) while achieving greater computational savings.

STOMP employs z-normalization (removing the mean and scaling by the standard deviation), making it sensitive to shape similarity while being invariant to amplitude differences. This differs fundamentally from our raw deviation-based approach, which preserves amplitude information (using min–max normalization). Therefore, we include STOMP primarily to illustrate alternative normalization strategies and their impact on the discovered insights rather than as a direct competitor in shift-based efficiency evaluation (as further illustrated in our first experiment in Figure 5).

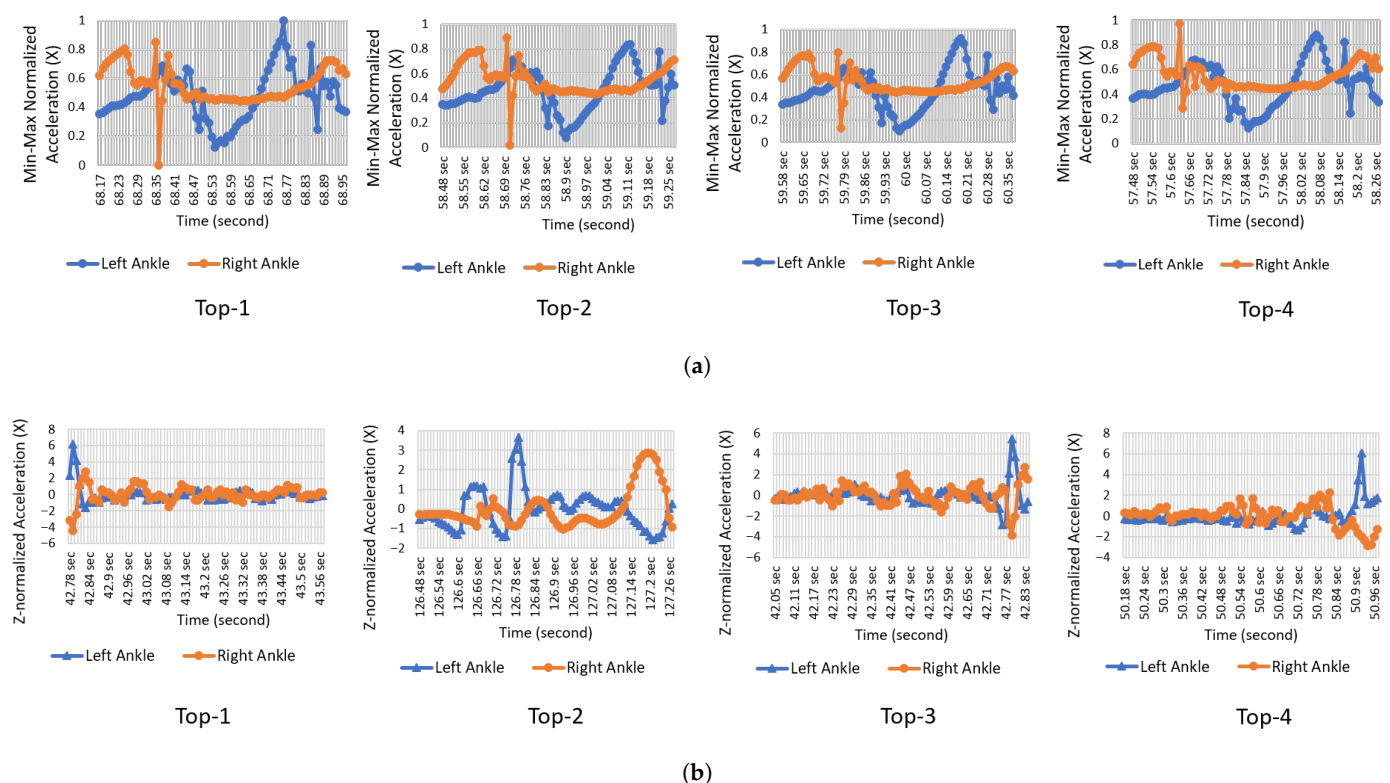


Figure 5. Comparison of top-4 dissimilar gait asymmetry insights between left and right ankle sensors from the WearGait-PD dataset using subsequence length $R = 80$. (a) Our proposed TiVEx-hybrid method. (b) STOMP method.

Experimental Setup. All experiments were conducted on a machine equipped with an Intel(R) Core(TM) i7-9750H CPU @ 2.60 GHz and 16 GB of RAM, using Python 3.9. To ensure reliability, each experiment measuring execution time was independently repeated 10 times, and the results were averaged, showing consistent trends across all runs.

8. Experimental Results and Analysis

In this section, we evaluate TiVEx across three key dimensions: computational efficiency, solution effectiveness, and practical applicability. We begin by comparing insight discovery capabilities between TiVEx-hybrid and STOMP on a clinical gait analysis task, demonstrating how normalization choices affect the types of insights discovered. We then present a quantitative performance analysis across varying parameter configurations and dataset characteristics.

Impact of Normalization on Discovered Insights. To illustrate how the normalization choice affects the types of insights discovered, we compare the top-4 dissimilar subsequence pairs identified by TiVEx-hybrid (using min–max normalization) versus STOMP (using z-normalization) on the WearGait-PD dataset with $R = 80$ and $S = 1$. Both methods enforce an exclusion zone of $R/2$ to ensure the temporal separation of discovered insights.

As shown in Figure 5a, TiVEx-hybrid with min–max normalization discovers subsequence pairs exhibiting large amplitude deviations—regions where the left and right ankle signals show substantial differences in movement magnitude. The top-1 view (68.17–68.96 s) demonstrates a crossing pattern where the left and right ankle signals intersect and then diverge, while other views show different amplitude relationships and temporal dynamics between the limb movements. These insights are clinically relevant for detecting gait asymmetries, as they highlight moments when limb movements show substantial magnitude differences.

In contrast, STOMP with z-normalization identifies subsequence pairs with shape-based dissimilarities, including flat, low-variance regions where signals are nearly constant but exhibit minor fluctuations (top-1, top-3, and top-4 views in Figure 5b). This occurs because z-normalization divides by the standard deviation, which is very small in flat regions. This small divisor amplifies minor fluctuations, causing these segments to rank highly based on shape dissimilarity [51]. While this behavior is appropriate for STOMP's design goal of shape dissimilarity, such flat-region discoveries provide limited value for gait asymmetry analysis, where amplitude differences during active movement phases are more clinically informative.

This comparison demonstrates that the choice of normalization determines the types of insights that are discovered. Min–max normalization preserves absolute deviation magnitudes, making it suitable for detecting asymmetric limb motion where scale matters. Z-normalization enables scale-invariant shape matching, making it suitable for finding unusual waveform patterns regardless of amplitude. For cross-dissimilarity exploration in physiological monitoring, min–max normalization uncovers insights that are better aligned with clinical interpretation needs.

We now present a systematic evaluation of computational performance across our proposed techniques and established baselines using the Google Mobility dataset. The experiment shown in Figure 6 demonstrates the computational efficiency of our proposed techniques—TiVEx-pruning and TiVEx-hybrid—across varying subsequence lengths (R), in comparison with an exhaustive search, TiVEx-sharing, and STOMP. The evaluation is conducted on the Google Mobility dataset with a shift length of $S = 1$. To clearly visualize differences in computational efficiency, the y-axis is plotted on a logarithmic scale. As expected, execution time increases for all methods with larger R values due to the quadratic growth in pairwise distance computations.

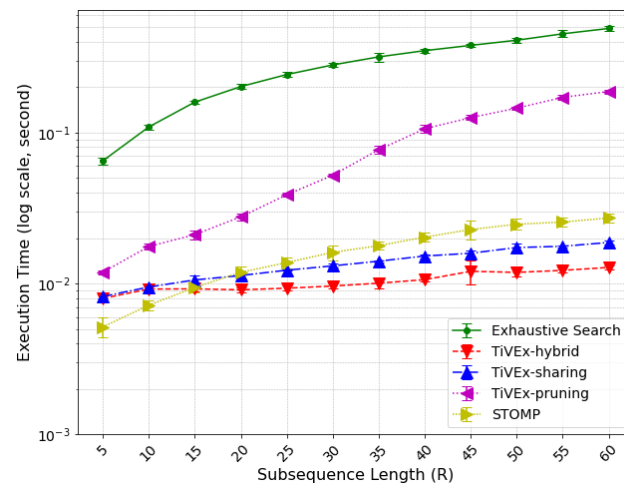


Figure 6. Execution time (log scale, in seconds) of all evaluated techniques across varying subsequence lengths (R), measured on the Google Mobility dataset with shift length $S = 1$.

The exhaustive search method, which evaluates all possible candidates without any optimization, incurs the highest execution time, reaching approximately 0.5 s at $R = 60$. For STOMP, the reported execution time includes both (1) the full Matrix Profile computation using standard STOMP incremental calculations and (2) searching and sorting of the main diagonal entries (aligned pairs). This measurement reflects the complete computational cost of using STOMP to identify top- k dissimilar aligned pairs. STOMP achieves moderate efficiency (0.0055–0.0287 s) through optimized incremental calculations but cannot incorporate pruning due to the completeness requirements of the Matrix Profile. TiVEx-sharing demonstrates improved performance over both baselines (0.0095–0.0187 s) by exploiting computation sharing among overlapping subsequences. However, TiVEx-sharing lacks pruning capabilities, limiting its efficiency gains compared to our TiVEx-hybrid approach.

TiVEx-hybrid delivers superior performance across different subsequence lengths (0.0079–0.012 s), achieving approximately a $2.3\times$ speedup over STOMP and a $38.5\times$ speedup over exhaustive search at $R = 60$. TiVEx-pruning shows higher execution times (0.012–0.18 s) than TiVEx-hybrid across all R values, demonstrating that the combination of computation-sharing and pruning strategies in TiVEx-hybrid produces benefits that exceed those of either optimization approach individually.

These computational efficiency results demonstrate the clear advantages of TiVEx-hybrid and motivate further evaluation across varying shift and subsequence lengths. STOMP is excluded from subsequent experiments for two reasons: (1) it is optimized specifically for $S = 1$, making comparisons with our shift-aware methods methodologically inconsistent, and (2) it cannot support time-shifted exploration ($S > 1$) due to its algorithmic design, whereas time-shifted analysis is central to our TiVEx methods.

Impact of Varying Shift Length S on Computational Efficiency. In the following experiment, we conduct an assessment of computational efficiency (in terms of the number of operations required and saved operations) achieved by our proposed approaches compared to both our previous technique and baseline approaches. We analyze the number of operations and cost savings across different shift lengths while keeping the subsequence length and k value constant (i.e., $R = 24$ and $k = 4$). Only shift lengths that produce an integer number of views are considered.

To emphasize the general impact of shift length S on computational burden, Figure 7a presents the absolute number of operations required by each method across varying shift lengths. This figure reflects the reduction in candidate set size as S increases (fewer views

to evaluate). As shown in Figure 7a, exhaustive search requires the most operations since it recalculates distances for each view independently. Increasing shift length reduces operations for all techniques due to fewer views and less overlap between successive views. Despite diminishing cost disparity at higher shift lengths, our methods significantly outperform exhaustive search, with approximately 18%, 31%, and 33% savings at $S = 12$ for TiVEx-pruning, TiVEx-sharing, and TiVEx-hybrid, respectively.

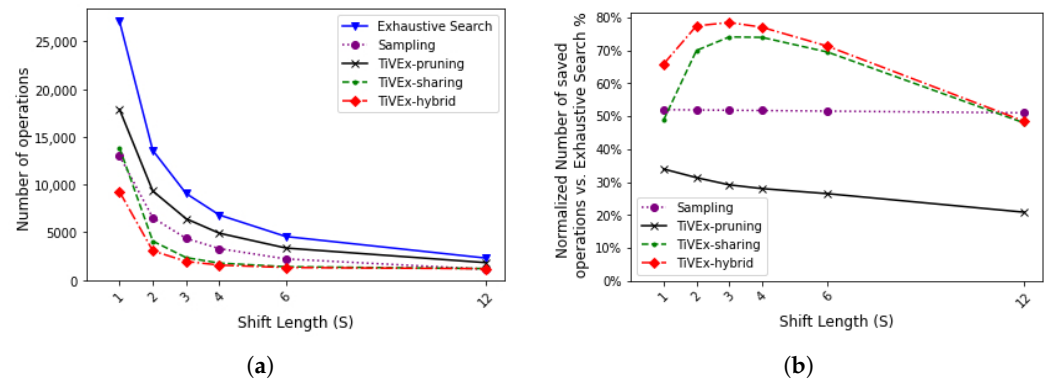


Figure 7. Comparison of the computational cost on the Google Mobility dataset with default parameters while varying S . (a) Number of operations. (b) Normalized number of saved operations vs. Exhaustive Search.

The convergence at higher shift lengths reflects a fundamental principle: as temporal separation increases, opportunities for both computation sharing and pruning diminish. At $S = 12$, the minimal overlap means that most calculations must be computed independently, explaining why the performance gap narrows. Nevertheless, TiVEx-hybrid consistently outperforms the individual techniques across all shift lengths. Even with the sampling rate $r_s = 0.5$, both TiVEx-sharing and TiVEx-hybrid maintain lower computational costs.

To isolate the impact of our algorithmic optimization and eliminate the effect of candidate set size reduction, Figure 7b presents normalized savings defined as the ratio of saved operations to total operations in exhaustive search. This normalization allows us to focus on the optimization benefits independent of how many views are evaluated. Both TiVEx-sharing and TiVEx-hybrid techniques exhibit substantial savings, particularly for smaller values of S , with the largest savings of approximately 78.5% for TiVEx-hybrid at $S = 3$. However, as S increases, the savings decrease, indicating a reduction in the number of overlapping points between successive views. This shift-length dependency highlights a key trade-off: our methods excel for fine-grained temporal analysis (small S) but offer diminishing advantages for coarse-grained analysis (large S), where sampling's consistent 50% reduction becomes competitive.

Moreover, the savings in both TiVEx-sharing and TiVEx-hybrid techniques surpass TiVEx-pruning, reflecting the latter's dependence on data characteristics that enable effective pruning—specifically, the presence of unpromising candidate views that can eliminate more computations early. When data lacks such distinguishable insights, the opportunities for pruning are significantly reduced, thereby limiting the effect of TiVEx-pruning. We provide a detailed analysis of this data-dependent behavior using controlled synthetic datasets to further validate our findings.

The normalized saved operations for the sampling approach remain consistently around 50% across all shift lengths when using a sampling rate of 0.5. It outperforms TiVEx-pruning across all shift lengths. However, both TiVEx-sharing and TiVEx-hybrid achieve greater savings across most shift lengths, with the exception of $S = 12$, where their performance slightly falls below that of sampling due to reduced overlap opportunities.

This variation can be attributed to the characteristics of the data, highlighting the inherent trade-off between the efficiency and effectiveness of the discovered visualizations—a point we further explore in the following experiment.

Effectiveness Analysis. As expected, TiVEx-pruning, TiVEx-sharing, and TiVEx-hybrid produce identical top- k views under fixed parameters since they employ different computational strategies to achieve the same final results. Figure 8 presents how the average deviation varies with increasing k values for different shift lengths using the average deviation metric (see Section 7).

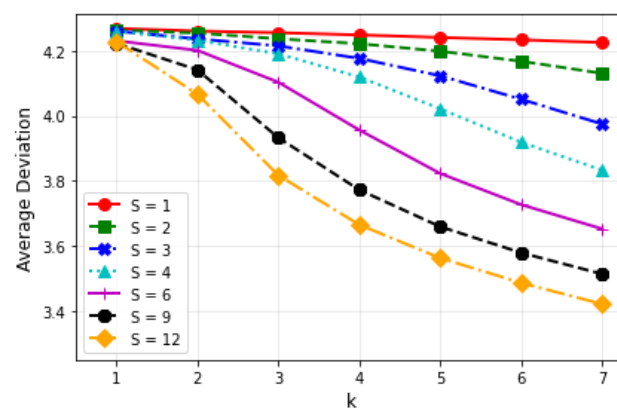


Figure 8. Measuring effectiveness using the average deviation metric on the Google Mobility dataset with default parameters while varying k and S .

Figure 8 reveals a consistent downward trend in average deviation as k increases across all shift lengths. This occurs because as we consider more views (larger k), we include progressively lower-ranked views with smaller deviation values, thereby reducing the overall average. This pattern is expected and demonstrates the natural ordering of views by their deviation scores.

More importantly, we observe how different shift lengths compare to the baseline $S = 1$ (red line). The baseline consistently achieves the highest average deviation across all k values, as it explores all possible view positions without skipping any candidates. At $S = 2$ (green line), the average deviation remains remarkably close to the baseline, with only a marginal gap. For example, at $k = 4$, $S = 2$ achieves approximately 98% of the baseline's average deviation, demonstrating that exploring every other position captures nearly the same quality of insights as the exhaustive search. At $S = 3$ (blue line), the gap widens slightly but remains small, maintaining around 95–96% of the baseline quality across all k values.

As the shift length increases further, the separation becomes more pronounced. At $S = 6$ (purple line), the average deviation drops to approximately 90–92% of the baseline, while $S = 9$ (black line) and $S = 12$ (orange line) achieve roughly 85–88% and 82–85% of baseline quality, respectively. Despite these reductions, the average deviations remain substantially high in absolute terms, indicating that even sparse exploration with large shift values successfully identifies highly dissimilar views.

These results demonstrate a clear trade-off between exploration granularity and insight quality. While larger shift values reduce computational cost by evaluating fewer candidate positions, they also slightly decrease the average deviation of discovered views. However, the degradation is graceful and predictable: doubling the shift length from $S = 2$ to $S = 4$ reduces operations from 63% to 84% savings compared to $S = 1$ (Figure 7), while the average deviation decreases by only an additional 2–3%.

TiVEx-Hybrid vs. Sampling: Efficiency–Effectiveness Trade-Offs. Given the superior performance of TiVEx-hybrid, it is selected to compare efficiency and effectiveness

with the sampling technique under varying sampling rates (i.e., $r_s = [0.2, 0.4, 0.5, 0.6, 0.8]$). Figure 9a shows that, with a sampling rate of $r_s = 0.5$, TiVEx-hybrid requires significantly fewer operations than sampling across all shift lengths. However, the efficiency of the sampling method is directly influenced by the chosen sampling rate, which in turn affects the quality of the results (i.e., effectiveness). To better understand this trade-off, we investigate the performance of the sampling technique under different sampling rates, comparing it with our TiVEx-hybrid technique.

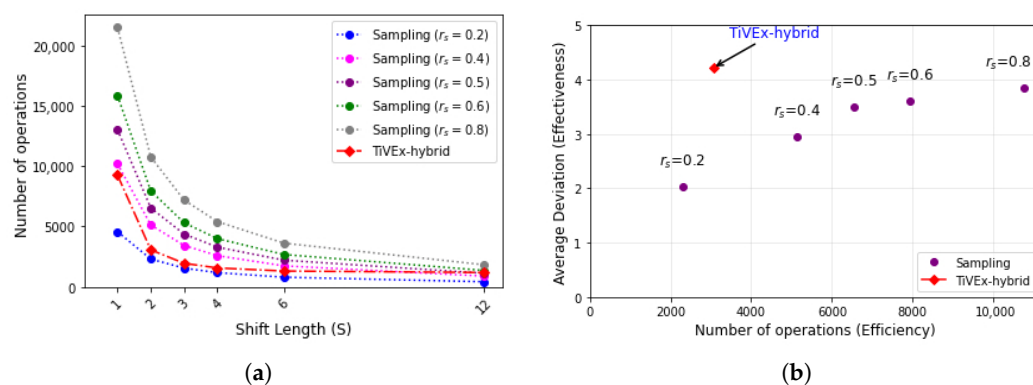


Figure 9. Trade-offs between TiVEx-hybrid and sampling in terms of efficiency and effectiveness on the Google Mobility dataset with default parameters. (a) Impact of shift length (S) on the number of operations using different sampling rates. (b) Efficiency–effectiveness trade-off for different sampling rates and a fixed shift length $S = 2$.

Figure 9b illustrates the trade-off between computational cost and the quality of the discovered visualizations across different sampling rates. For the sampling technique, smaller sampling rates (e.g., $r_s = 0.2$) result in higher efficiency, as fewer operations are required. However, this comes at the cost of reduced effectiveness, as indicated by lower effectiveness values due to the diminished quality of the discovered visualizations. As the sampling rates increase, efficiency decreases while effectiveness improves, with the number of operations ranging from 2303 to 10,763 and average deviation values increasing from 2 to 3.8. In contrast, TiVEx-hybrid achieves a balanced trade-off, requiring approximately 3060 operations while maintaining a high deviation value close to 4.3. This demonstrates that TiVEx-hybrid effectively minimizes computational costs while preserving high-quality visualizations, outperforming sampling across all sampling rates.

Quality of Discovered Insights. While the efficiency–effectiveness scatter plot demonstrates the computational advantages of TiVEx-hybrid, we verify that this efficiency does not compromise the quality of the discovered insights. Figure 10 presents the top-4 most dissimilar views discovered by exhaustive search (right), sampling ($r_s = 0.5$, middle), and TiVEx-hybrid (left) on the Google Mobility dataset (workplace vs. residential for Brazil).

The results demonstrate that TiVEx-hybrid discovers visually identical views to exhaustive search with identical deviation values ($D = [3.79, 3.72, 3.71, 3.66]$). Examining each ranking, both methods identify the same crossing patterns between workplace and residential mobility at the same temporal positions (e.g., Top-1 from 2 November 2021 to 25 November 2021). This temporal and behavioral alignment across all four rankings provides empirical proof of exact result optimality.

In contrast, the sampling technique discovers different views. While Top-1 through Top-3 show similar crossing patterns to both TiVEx-hybrid and exhaustive search, they occur at shifted temporal positions due to downsampling artifacts. Notably, the average deviation of sampling’s top-4 views is lower than that of TiVEx-hybrid and exhaustive search, indicating that sampling discovers views with weaker dissimilarity overall. Most importantly, sampling’s top-4 view exhibits the opposite behavior: residential mobility

risers steeply while workplace mobility decreases. This is qualitatively different from the true top-4 most dissimilar views identified by both TiVEx-hybrid and exhaustive search, demonstrating that downsampling destroyed the fine-grained temporal information needed for accurate insight discovery.

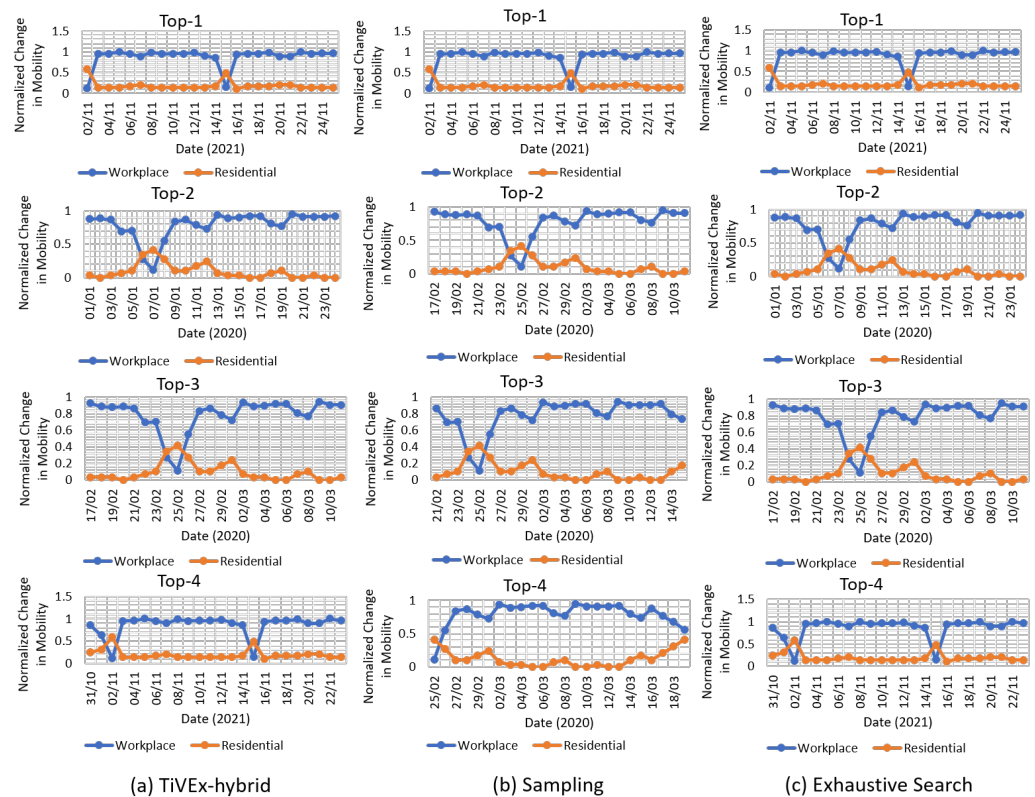


Figure 10. Top-4 dissimilar views discovered by (a) TiVEx-hybrid, (b) sampling, and (c) exhaustive search methods on the Google Mobility dataset and a shift length $S = 2$.

Impact of Varying Subsequence Length R on Computational Efficiency. Similarly, the experiments are conducted to assess the computational efficiency achieved by our techniques, considering different subsequence length values while keeping the shift length and k value constant (i.e., $S = 6$ and $k = 4$). To ensure a fair comparison, we focus on subsequence lengths that yield an integer number of views. The result of this experiment is shown in Figure 11 using the Google mobility dataset.

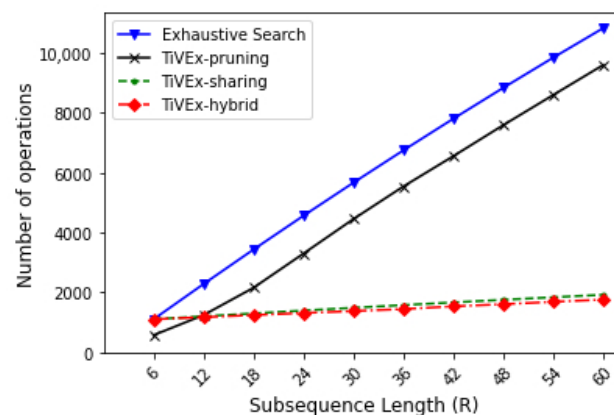


Figure 11. Comparison of computational cost on the Google Mobility dataset with default parameters while varying R .

As shown in Figure 11, the results clearly highlight a recognizable difference in the number of operations required between exhaustive search and other techniques. As the subsequence length increases, all techniques show higher computational costs due to the larger number of data points in each view, leading to more intensive pairwise distance calculations. The disparity in operations between exhaustive search and other techniques significantly widens, with TiVEx-hybrid achieving approximately 84% savings compared to exhaustive search at $R = 60$, as depicted in Figure 11.

TiVEx-pruning requires more operations than TiVEx-sharing and TiVEx-hybrid as the subsequence length increases. This is due to the sequential nature of how TiVEx-pruning is applied and dataset characteristics, such as the positioning of views with the largest deviation, which impacts the number of saved/pruned computations. However, at small subsequence lengths (e.g., $R = 6$), TiVEx-pruning outperforms other methods because (1) when $R = 6$ and $S = 6$, consecutive views have no overlap, eliminating TiVEx-sharing's advantages, and (2) TiVEx-pruning's fine-grained, point-by-point evaluation becomes more efficient than the pane-based strategy used in TiVEx-hybrid.

To assess the generalizability of these computational efficiency trends, we evaluate our techniques on three additional datasets with distinct characteristics: the stock market, random walk, and seismic datasets. Figure 12a–c demonstrate consistent performance rankings across all datasets, with TiVEx-hybrid consistently outperforming other methods. Computational savings compared to exhaustive search are approximately 84% (Google Mobility, $R = 60$), 72% (Stock Market, $R = 56$), 71% (Random Walk, $R = 58$), and 86% (Seismic, $R = 60$). While method rankings remain stable, the magnitude of efficiency gains varies with dataset characteristics, confirming the broad applicability of our approach across diverse temporal structures. For brevity, all subsequent experiments use the Google Mobility dataset, chosen for its relevance as a real-world time series.

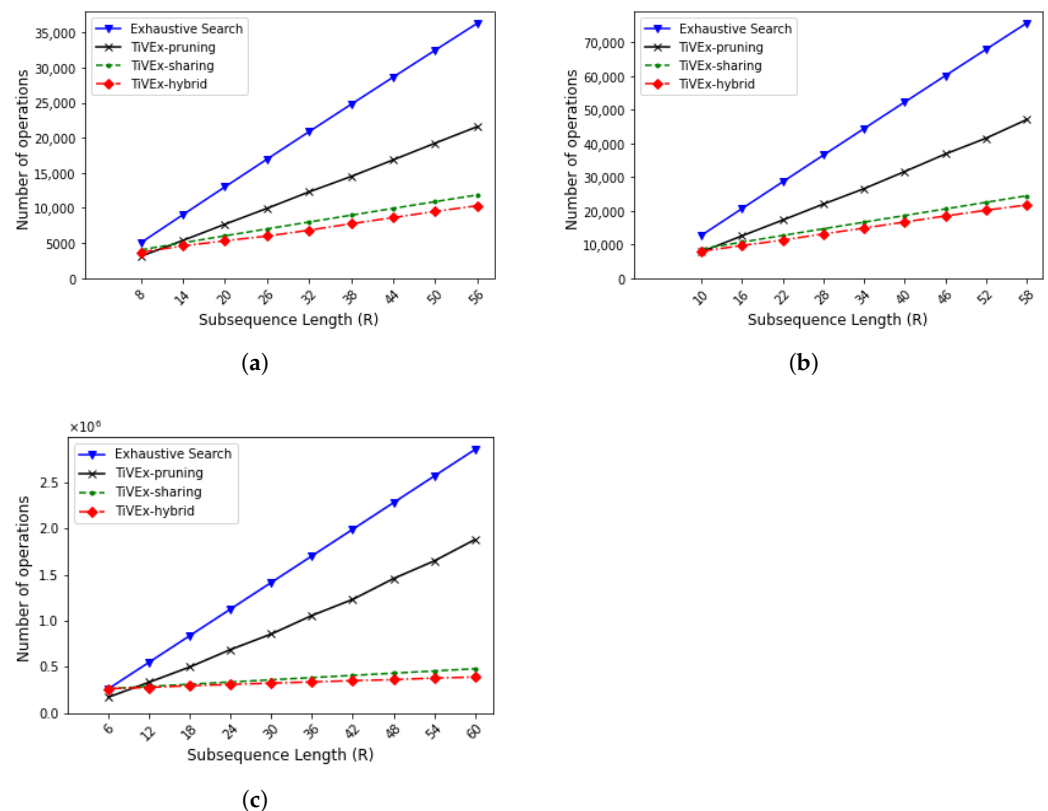


Figure 12. Comparison of computational cost across three datasets with default parameters while varying R . (a) Stock market dataset ($L = 2012$). (b) Random walk dataset ($L = 4000$). (c) Seismic dataset ($L = 144,000$).

An analysis of our real-world datasets (Google Mobility and stock market) reveals that top- k insights exhibit dramatically varying temporal distributions. The stock market dataset ($L = 2012$, $R = 26$, $S = 6$) shows extremely clustered insights, where the top-4 insights appear in consecutive positions, spanning only 0.9% of the exploration sequence. In contrast, the Google Mobility dataset ($L = 600$, $R = 24$, $S = 6$) shows extremely distributed insights, where the top-4 insights appear at the beginning and end, spanning 99% of the total views. Despite these vastly different temporal distributions, TiVEx-hybrid achieves computational savings of 71% for Google Mobility and 65% for the stock market compared to exhaustive search (as shown in Figure 11 and Figure 12a, respectively). This demonstrates that the method maintains consistent efficiency across different insight distributions.

Impact of Deviation Position on Computational Efficiency. A fundamental question for any pruning-based optimization is whether efficiency depends on the distribution of high-utility items in the search space. To characterize this algorithmic property, we conduct a controlled experiment using a synthetic sine–cosine dataset, where the largest deviation is systematically positioned at different locations (beginning, middle, and end), as shown in Figure 13(left). This deviation placement allows us to establish performance bounds independent of other dataset characteristics.

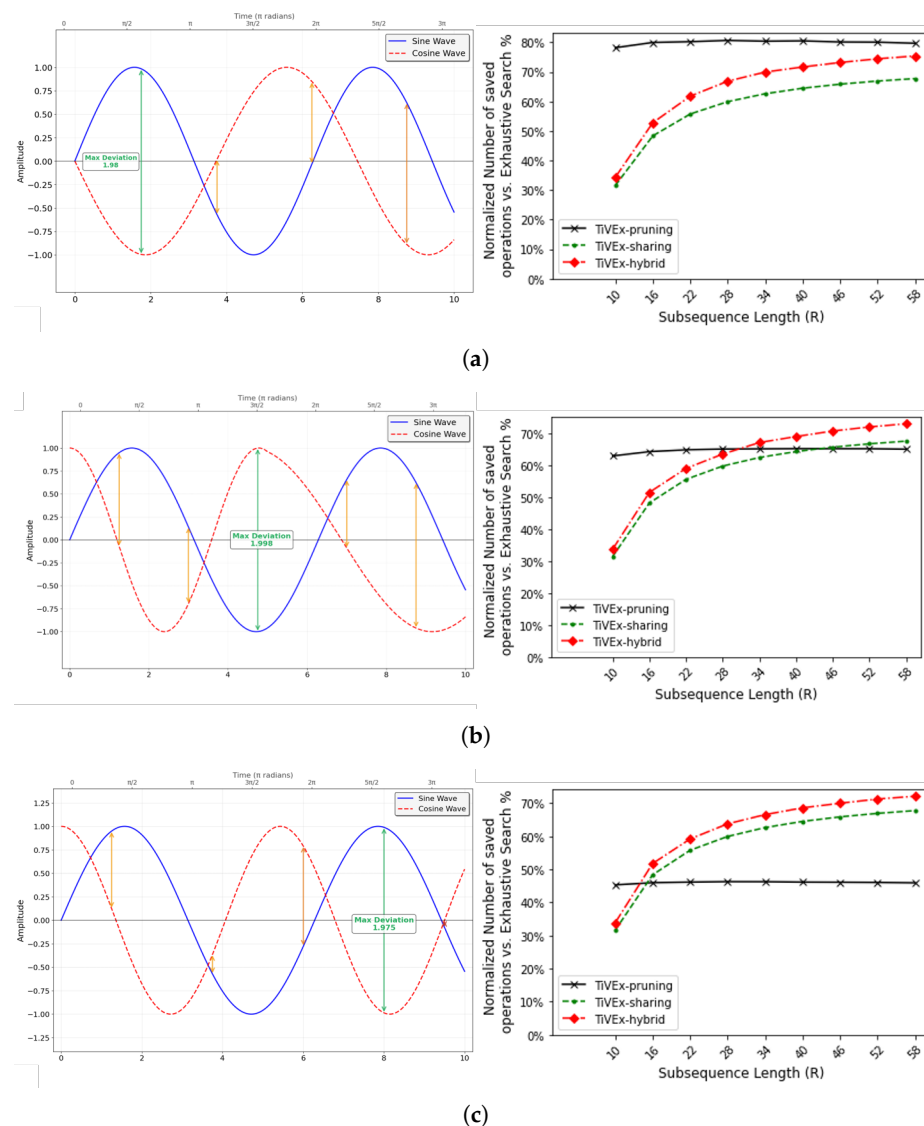


Figure 13. Impact of deviation placement on normalized savings using synthetic sine–cosine data. Each scenario uses a dataset with default parameters and $L = 4000$ while varying R . (a) Largest deviation at the beginning. (b) Largest deviation in the middle. (c) Largest deviation at the end.

When the views with the largest deviation occur at the beginning (Figure 13a), TiVEx-pruning achieves significant computational savings of approximately 80%, as the algorithm can quickly identify and discover the high-deviation insight early in the exploration process. This early identification allows subsequent candidates to prune computations more aggressively, as they can be compared against a strong reference deviation threshold. In this scenario, TiVEx-pruning outperforms TiVEx-hybrid, since its fine-grained, point-level pruning is more effective than the pane-level pruning employed in TiVEx-hybrid.

Conversely, when the views with the largest deviation appear in the middle (Figure 13b) or at the end (Figure 13c), the algorithm must evaluate more candidates before finding the optimal insight (with the largest deviation), thereby reducing pruning opportunities. Performance decreases when the deviation is positioned in the middle (65% savings) and drops further when located at the end (45% savings). This behavior stems from TiVEx-pruning's sequential evaluation strategy: early detection of high-deviation insights enables more aggressive pruning of unpromising candidate views.

In contrast, TiVEx-hybrid maintains more consistent performance across all deviation positions (72–75% savings), demonstrating the stability provided by combining sharing and pruning strategies. TiVEx-sharing, as expected, shows consistent savings across all scenarios since it relies solely on view overlap, which remains constant regardless of the position of deviation.

These controlled experiments establish the expected performance range for TiVEx-pruning (45–80% savings) under different deviation distributions. As illustrated in Figures 11 and 12a,b, our real-world datasets exhibit naturally distributed deviation insights, with observed savings falling within this characterized range, validating that the controlled experiments may predict real-world performance variability.

The efficiency of TiVEx-pruning is data-dependent. In the worst case, where all views have similarly high deviations and no early pruning triggers, TiVEx-pruning must evaluate all point pairs in all views, approaching the cost of the exhaustive search. However, this worst-case scenario is rare in practice: even when the largest deviation appears at the end of the time series (Figure 13c), partial pruning still achieves 45% savings.

TiVEx-hybrid provides robustness through its integrated design, though per-view effectiveness depends on pane configuration. When overlapping panes have been processed (Cases 1 and 3 in Figure 4), sharing provides efficiency gains through pane reuse, establishing a performance floor even when pruning opportunities are minimal. However, when all overlapping panes are unseen (Case 2), TiVEx-hybrid relies solely on pruning for that view. While individual views in worst-case configurations (Case 2 with no pruning) may approach the exhaustive search cost, this scenario rarely persists across multiple consecutive views, ensuring strong overall performance (72–75% average savings, Figure 13).

Impact of Varying k Values on Computational Efficiency. In this experiment, we assess efficiency across different k values while maintaining fixed parameters ($R = 24$, $S = 6$). As shown in Figure 14, exhaustive search and TiVEx-sharing maintain constant computational costs regardless of k values, since their algorithms are independent of the result set size (i.e., top- k).

However, the TiVEx-pruning method exhibits different behavior. It shows significant performance degradation, with operations increasing from approximately 2200 at $k = 1$ to 3600 at $k = 7$ ($\approx 64\%$ increase). This occurs because larger k values require maintaining more candidates in the top- k set, making pruning thresholds less restrictive and reducing opportunities for early candidate elimination. In contrast, TiVEx-hybrid demonstrates remarkable stability, with operations increasing only slightly from 1200 to 1400 ($\approx 17\%$ increase). This stability results from the hybrid approach: while the pruning component

becomes less effective with larger k , the sharing component provides consistent baseline efficiency regardless of result set size.

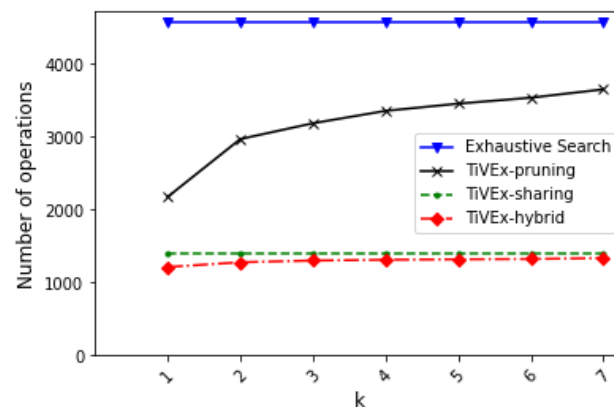


Figure 14. Comparing the computational cost on the Google Mobility dataset with default parameters while varying k .

Summary of Experimental Findings. Our experimental evaluation across six datasets reveals that TiVEx-hybrid consistently achieves the best performance, providing 72–84% computational savings while maintaining exact result optimality through its integration of computation sharing and pruning. For parameter selection, shift length S presents a clear efficiency–effectiveness trade-off, where $S = 2$ – 3 maintains 95–98% of baseline quality with 63–78.5% savings, while $S = 6$ – 12 maximizes efficiency (84% savings) with graceful quality degradation (82–90% of baseline); subsequence length R primarily affects computational cost and should be chosen based on domain knowledge; and $k = 3$ – 5 provides sufficient insight diversity with stable TiVEx-hybrid performance. Compared to other existing methods, TiVEx-hybrid achieves $2.3\times$ speedup over STOMP and dominates sampling-based approaches across all configurations. TiVEx-hybrid also demonstrates robustness across different deviation distributions (72–75% consistent savings), whereas TiVEx-pruning exhibits data-dependent performance (45–80% range) that excels when high-deviation insights appear early in the exploration sequence. These findings enable practitioners to select parameters as follows: for shift length S , use $S = 1$ for comprehensive exploration benefiting from algorithmic optimization, or larger S values when computational constraints necessitate trading temporal quality for speed; for subsequence length R , choose values based on domain-specific temporal insights of interest (e.g., hourly, daily, weekly); for k , use $k = 3$ – 5 to balance the discovery of diverse insights with computational efficiency, with larger k reserved for scenarios requiring extensive exploration.

9. Conclusions

Interactive visual exploration of time series data requires efficiently identifying the most informative subsequence comparisons from a combinatorially large search space. This paper proposes the TiVEx family of techniques—TiVEx-sharing, TiVEx-pruning, and TiVEx-hybrid—which enable the scalable discovery of top- k dissimilar subsequence views through strategic computation sharing and distance-based pruning. Building upon our previous work on TiVEx-sharing, we present two novel contributions: TiVEx-pruning, which leverages distance bounds to eliminate unpromising candidates early, and TiVEx-hybrid, which integrates both mechanisms for maximum efficiency. Our experimental evaluation on diverse real-world datasets demonstrated that TiVEx-hybrid achieves substantial computational savings—reducing operations by up to 84% compared to exhaustive search—while maintaining result quality to be exactly the same as in exhaustive exploration. The effectiveness analysis confirmed that even with shift lengths of $S > 1$, our methods discover views

with average deviations close to the optimal $S = 1$ baseline, validating the practical utility of our approach for large-scale time series analysis. By balancing computational efficiency with insight quality, TiVEx empowers analysts to rapidly discover meaningful insights in complex temporal data across diverse domains, including healthcare monitoring, financial analysis, and sensor network analytics. While the current implementation of TiVEx operates on static archival time series, several promising directions remain for future investigation. First, extending TiVEx to streaming scenarios would enable continuous insight discovery from evolving data streams, requiring adaptations to handle concept drift and maintain efficient incremental updates. Second, incorporating user feedback mechanisms could further refine the ranking of discovered views based on domain-specific preferences. Third, exploring multi-resolution exploration strategies that adaptively adjust shift length S based on local data characteristics could further optimize the exploration–efficiency trade-off. Finally, investigating parallel and distributed implementations of TiVEx-hybrid could unlock even greater scalability for massive-scale time series datasets.

Author Contributions: Conceptualization, H.H. and M.A.S.; Methodology, H.H. and M.A.S.; formal analysis and investigation, H.H. and M.A.S.; writing—original draft preparation, H.H.; writing—review and editing, M.A.S.; funding acquisition, M.A.S.; Resources, H.H. and M.A.S.; supervision, M.A.S.; approved—final manuscript, H.H. and M.A.S. All authors have read and agreed to the published version of the manuscript.

Funding: Partial financial support was received from the UAE University Strategic Research Program Grant No. 12R147.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Acknowledgments: This research is supported partially by UAE University Strategic Research Program Grant (12R147).

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. TiVEx-Sharing Detailed Example

This appendix provides detailed numerical examples to illustrate the TiVEx-sharing computation-sharing mechanism and cost analysis.

Appendix A.1. Applying TiVEx-Sharing to Time Series

Consider the example shown in Figure 2, where TiVEx-sharing is applied to a time series with a subsequence length $R = 8$ and a shift length $S = 6$. The pane size is calculated as $g = \gcd(8, 6) = 2$, meaning each subsequence is divided into panes of size 2.

For instance, the first view $v_{1,8}$ and the second view $v_{7,14}$ share overlapping panes at positions 7 and 8 (shown in the shaded area in Figure 2). By reusing the computations for these overlapping panes, TiVEx-sharing avoids redundant distance calculations.

Appendix A.2. Cost Calculation Example

Let us calculate the total cost for all three views in Figure 2 after applying the TiVEx-sharing technique.

The cost for the first view ($v_{1,8}$) includes all four panes, with a total length of 8:

1. Distance operations: Computing $(X_i - X'_i)^2$ for all 8 points costs 8 operations.

2. Aggregate operations within panes: Each pane of size $g = 2$ requires $(2 - 1) = 1$ aggregation operations. With $\frac{8}{2} = 4$ panes, this costs $1 \times 4 = 4$ operations.
3. Final aggregate operations: Combining 4 panes requires $(4 - 1) = 3$ operations.

Total cost for the first view: $8 + 4 + 3 = 15$ operations, which equals $2 \times 8 - 1 = 15$. For each subsequent view ($v_{7,14}$ and $v_{13,20}$), only the remaining partial aggregates (length $S = 6$) need to be computed:

1. Distance operations: Computing $(X_i - X'_i)^2$ for the 6 new points (e.g., positions 9–14) costs 6 operations.
2. Aggregate operations within the remaining panes: With 3 new panes of size $g = 2$, this costs $(2 - 1) \times 3 = 3$ operations.
3. Final aggregate operations for remaining panes: Combining 3 panes costs $(3 - 1) = 2$ operations.
4. Combining with overlapping aggregates: The 3 remaining panes must be combined with the $\frac{R-S}{g} = \frac{8-6}{2} = 1$ overlapping pane, requiring 1 additional operation.

Total cost for each subsequent view: $6 + 3 + 2 + 1 = 12$ operations, which equals $2 \times 6 - 1 + \frac{8-6}{2} = 12$.

Total Cost for All Three Views: Using TiVEx-sharing: $15 + 12 + 12 = 39$ operations.

Using exhaustive search (Equation (2)): $(2 \times 8 - 1) \times 3 = 15 \times 3 = 45$ operations. This demonstrates that the TiVEx-sharing achieves savings of $\frac{45-39}{45} \times 100\% = 13.3\%$ by reusing overlapping computations.

Appendix B. Proofs

Proof of Lemma 1. The true deviation for view $v_{b,e}$ is defined as in Equation (1). We can partition the positions of point pairs into processed (known) and unprocessed (unknown) ranges. Given that positions b to j have been processed, we can rewrite the deviation as

$$D(v_{b,e}) = \sqrt{\sum_{i=b}^j (X_i - X'_i)^2 + \sum_{i=j+1}^e (X_i - X'_i)^2}$$

Since all data points are normalized to $[0, 1]$ via min-max normalization, for any normalized values $X_i, X'_i \in [0, 1]$, the squared distance satisfies

$$(X_i - X'_i)^2 \leq \max_{a,b \in [0,1]} (a - b)^2 = (1 - 0)^2 = 1$$

The maximum value of 1 is achieved when one point equals 0 and the other equals 1. Applying this bound to all unprocessed positions from $j + 1$ to e

$$\sum_{i=j+1}^e (X_i - X'_i)^2 \leq \sum_{i=j+1}^e 1 = (e - j)$$

where $(e - j)$ is the number of unprocessed positions. Substituting this inequality into the deviation expression:

$$\begin{aligned} D(v_{b,e}) &= \sqrt{\sum_{i=b}^j (X_i - X'_i)^2 + \sum_{i=j+1}^e (X_i - X'_i)^2} \\ &\leq \sqrt{\sum_{i=b}^j (X_i - X'_i)^2 + (e - j)} \\ &= D_{\max}(v_{b,e}) \end{aligned}$$

The inequality holds because the square root function is monotonically increasing for non-negative arguments. Specifically, since

$$\sum_{i=b}^j (X_i - X'_i)^2 + \sum_{i=j+1}^e (X_i - X'_i)^2 \leq \sum_{i=b}^j (X_i - X'_i)^2 + (e - j)$$

and both sides are non-negative, taking the square root of both sides preserves the inequality.

Therefore, $D_{\max}(v_{b,e})$ is a valid upper bound for the true deviation $D(v_{b,e})$, i.e., $D(v_{b,e}) \leq D_{\max}(v_{b,e})$. $\square$

References

- Gogolou, A.; Tsandilas, T.; Palpanas, T.; Bezerianos, A. Comparing similarity perception in time series visualizations. *IEEE Trans. Vis. Comput. Graph.* **2018**, *25*, 523–533.
- Wang, X.; Mueen, A.; Ding, H.; Trajcevski, G.; Scheuermann, P.; Keogh, E. Experimental comparison of representation methods and distance measures for time series data. *Data Min. Knowl. Discov.* **2013**, *26*, 275–309.
- Rakthanmanon, T.; Campana, B.; Mueen, A.; Batista, G.; Westover, B.; Zhu, Q.; Zakaria, J.; Keogh, E. Addressing big data time series: Mining trillions of time series subsequences under dynamic time warping. *ACM Trans. Knowl. Discov. Data* **2013**, *7*, 10.
- Aghabozorgi, S.; Shirkhorshidi, A.S.; Wah, T.Y. Time-series clustering—A decade review. *Inf. Syst.* **2015**, *53*, 16–38.
- Sharaf, M.A.; Zhang, X.; Chrysanthi, P.K.; Alsaedi, W.; Alkalbani, M.; Helal, H.; Aldhaheri, A. CovidLens: Visually Understanding the COVID-19 Indicators through the Lens of Mobility Data. In *Proceedings of the 2022 23rd IEEE International Conference on Mobile Data Management (MDM)*, Paphos, Cyprus, 6–9 June 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 302–305. <https://doi.org/10.1109/MDM55031.2022.00066>.
- Keogh, E.; Lin, J.; Lee, S.H.; Herle, H.V. Finding the most unusual time series subsequence: Algorithms and applications. *Knowl. Inf. Syst.* **2007**, *11*, 1–27.
- Tableau Software. *Tableau Public*, Version 2025.3.2; Tableau Software: Seattle, WA, USA, 2025. Available online: <https://public.tableau.com> (accessed on 8 February 2026).
- Microsoft. *Power BI*, Version 2.150.2102.0; Microsoft: Redmond, WA, USA, 2026. Available online: <https://app.powerbi.com> (accessed on 8 February 2026).
- Qlik. *Qlik Sense*, Version Patch 2; Qlik Technologies Inc.: King of Prussia, PA, USA, 2025. Available online: <https://www.qlik.com> (accessed on 8 February 2026).
- Harris, C.; Rossi, R.A.; Malik, S.; Hoffswell, J.; Du, F.; Lee, T.Y.; Koh, E.; Zhao, H. Insight-Centric Visualization Recommendation. *arXiv* **2021**, arXiv:2103.11297.
- Shi, Y.; Chen, B.; Chen, Y.; Jin, Z.; Xu, K.; Jiao, X.; Gao, T.; Cao, N. Supporting guided exploratory visual analysis on time series data with reinforcement learning. *IEEE Trans. Vis. Comput. Graph.* **2023**, *30*, 1172–1182.
- Ceneda, D.; Gschwandtner, T.; May, T.; Miksch, S.; Schulz, H.J.; Streit, M.; Tominski, C. Characterizing guidance in visual analytics. *IEEE Trans. Vis. Comput. Graph.* **2016**, *23*, 111–120.
- Sacha, D.; Kraus, M.; Keim, D.A.; Chen, M. Vis4ml: An ontology for visual analytics assisted machine learning. *IEEE Trans. Vis. Comput. Graph.* **2018**, *25*, 385–395.
- Battle, L.; Heer, J. Characterizing exploratory visual analysis: A literature review and evaluation of analytic provenance in Tableau. *Comput. Graph. Forum* **2019**, *38*, 145–159. <https://doi.org/10.1111/cgf.13678>.
- Hu, W. Bridging the gap: An integrated approach to motif discovery and discord detection in time-series data. *Neurocomputing* **2025**, *619*, 129056.
- Toyoda, M.; Sakurai, Y. Discovery of Cross-Similarity in Data Streams. In *Proceedings of the 2010 IEEE 26th International Conference on Data Engineering (ICDE)*, Long Beach, CA, USA, 1–6 March 2010; IEEE: Piscataway, NJ, USA, 2010; pp. 101–104.
- Giao, B.C. Similarity join over multiple time series under Dynamic Time Warping. *CTU J. Innov. Sustain. Dev.* **2023**, *15*, 1–11.
- Mueen, A.; Hamooni, H.; Estrada, T. Time Series Join on Subsequence Correlation. In *Proceedings of the 2014 IEEE International Conference on Data Mining (ICDM)*, Shenzhen, China, 14–17 December 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 450–459.
- Helal, H.; Sharaf, M.A.; Masud, M.M.; Chrysanthi, P.K. TiVEx: Optimized Processing for Time Series Visual Exploration. In *Proceedings of the 2023 20th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA)*, Giza, Egypt, 4–7 December 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 1–8. <https://doi.org/10.1109/AICCSA59173.2023.10479252>.
- Yeh, C.C.M.; Zhu, Y.; Ulanova, L.; Begum, N.; Ding, Y.; Dau, H.A.; Silva, D.F.; Mueen, A.; Keogh, E. Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets. In *Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM)*, Barcelona, Spain, 12–15 December 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1317–1322. <https://doi.org/10.1109/ICDM.2016.0179>.

21. Synapse. *WearGait-PD Dataset*, Version 10.1101/2024.09.11.24313476; SAGE Bionetworks: Seattle, WA, USA, 2024. Available online: <https://www.synapse.org/Synapse:syn52540892/wiki/623751> (accessed on 8 February 2026).
22. Yahoo Finance; Yahoo Inc.: New York, NY, USA, 2025. Available online: <https://finance.yahoo.com/> (accessed on 8 February 2026).
23. Park, H.; Shin, S.; Youm, C.; Cheon, S.M. Deep learning-based detection of affected body parts in Parkinson's disease and freezing of gait using time-series imaging. *Sci. Rep.* **2024**, *14*, 23732.
24. Zhu, Y.; Zimmerman, Z.; Senobari, N.S.; Yeh, C.C.M.; Funning, G.; Mueen, A.; Brisk, P.; Keogh, E. Matrix Profile II: Exploiting a Novel Algorithm and GPUs to Break the One Hundred Million Barrier for Time Series Motifs and Joins. In *Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM), Barcelona, Spain, 12–15 December 2016*; IEEE: Piscataway, NJ, USA, 2016; pp. 739–748. <https://doi.org/10.1109/ICDM.2016.0085>.
25. Lian, X.; Chen, L. Efficient similarity join over multiple stream time series. *IEEE Trans. Knowl. Data Eng.* **2009**, *21*, 1544–1558.
26. Lin, Y.; McCool, M.D.; Ghorbani, A.A. Time series motif discovery and anomaly detection based on subseries join. *IAENG Int. J. Comput. Sci.* **2010**, *37*, 259–271.
27. Pan, S.; Wang, P.; Wang, C.; Wang, W.; Wang, J. NLC: Search correlated window pairs on long time series. *Proc. VLDB Endow.* **2022**, *15*, 1363–1375.
28. Li, J.; Maier, D.; Tufte, K.; Papadimos, V.; Tucker, P.A. No pane, no gain: Efficient evaluation of sliding-window aggregates over data streams. *ACM SIGMOD Rec.* **2005**, *34*, 39–44.
29. Linardi, M.; Zhu, Y.; Palpanas, T.; Keogh, E. Matrix Profile X: VALMOD-Scalable Discovery of Variable-Length Motifs in Data Series. In *Proceedings of the 2018 ACM International Conference on Management of Data (SIGMOD), Houston, TX, USA, 10–15 June 2018*; ACM: New York, NY, USA, 2018; pp. 1053–1066. <https://doi.org/10.1145/3183713.3183744>.
30. Linardi, M.; Zhu, Y.; Palpanas, T.; Keogh, E. Matrix profile goes MAD: Variable-length motif and discord discovery in data series. *Data Min. Knowl. Discov.* **2020**, *34*, 1022–1071.
31. Yeh, C.C.M.; Zhu, Y.; Ulanova, L.; Begum, N.; Ding, Y.; Dau, H.A.; Zimmerman, Z.; Silva, D.F.; Mueen, A.; Keogh, E. Time series joins, motifs, discords and shapelets: A unifying view that exploits the matrix profile. *Data Min. Knowl. Discov.* **2018**, *32*, 83–123.
32. Google. *COVID-19 Community Mobility Reports*; Google LLC: Menlo Park, CA, USA, 2020. Available online: <https://www.google.com/covid19/mobility> (accessed on 8 February 2026).
33. Vartak, M.; Rahman, S.; Madden, S.; Parameswaran, A.; Polyzotis, N. SeeDB: Efficient data-driven visualization recommendations to support visual analytics. *Proc. VLDB Endow.* **2015**, *8*, 2182–2193.
34. Ding, R.; Han, S.; Xu, Y.; Zhang, H.; Zhang, D. QuickInsights: Quick and Automatic Discovery of Insights from Multi-Dimensional Data. In *Proceedings of the 2019 ACM International Conference on Management of Data (SIGMOD), Amsterdam, The Netherlands, 30 June–5 July 2019*; ACM: New York, NY, USA, 2019; pp. 317–332. <https://doi.org/10.1145/3299869.3314037>.
35. Ehsan, H.; Sharaf, M.A.; Chrysanthis, P.K. MUVE: Efficient Multi-Objective View Recommendation for Visual Data Exploration. In *Proceedings of the 2016 IEEE 32nd International Conference on Data Engineering (ICDE), Helsinki, Finland, 16–20 May 2016*; IEEE: Piscataway, NJ, USA, 2016; pp. 731–742. <https://doi.org/10.1109/ICDE.2016.7498285>.
36. Ehsan, H.; Sharaf, M.A.; Chrysanthis, P.K. Efficient Recommendation of Aggregate Data Visualizations. *IEEE Trans. Knowl. Data Eng.* **2017**, *30*, 263–277. <https://doi.org/10.1109/TKDE.2017.2765634>.
37. Sharaf, M.A.; Mafrur, R.; Zuccon, G. Efficient Diversification for Recommending Aggregate Data Visualizations. *IEEE Access* **2023**, *11*, 62261–62280. <https://doi.org/10.1109/ACCESS.2023.3283457>.
38. Lu, Y.; Srinivas, T.V.A.; Nakamura, T.; Imamura, M.; Keogh, E. Matrix Profile XXX: MADRID: A Hyper-Anytime and Parameter-Free Algorithm to Find Time Series Anomalies of All Lengths. In *Proceedings of the 2023 IEEE International Conference on Data Mining (ICDM), Shanghai, China, 1–4 December 2023*; IEEE: Piscataway, NJ, USA, 2023; pp. 1199–1204. <https://doi.org/10.1109/ICDM58522.2023.00148>.
39. Nakamura, T.; Mercer, R.; Imamura, M.; Keogh, E. MERLIN++: Parameter-free discovery of time series anomalies. *Data Min. Knowl. Discov.* **2023**, *37*, 670–709.
40. Camerra, A.; Palpanas, T.; Shieh, J.; Keogh, E. iSAX 2.0: Indexing and Mining One Billion Time Series. In *Proceedings of the 2010 IEEE International Conference on Data Mining (ICDM), Sydney, Australia, 13–17 December 2010*; IEEE: Piscataway, NJ, USA, 2010; pp. 58–67. <https://doi.org/10.1109/ICDM.2010.124>.
41. Abdi, H.; Williams, L.J. Principal component analysis. *WIREs Comput. Stat.* **2010**, *2*, 433–459.
42. Ghafari, J.; Jassim, S. Singular-Value-Decomposition-Based Matrix Surgery. *Entropy* **2024**, *26*, 701.
43. Ding, H.; Trajcevski, G.; Scheuermann, P.; Wang, X.; Keogh, E. Querying and mining of time series data: Experimental comparison of representations and distance measures. *Proc. VLDB Endow.* **2008**, *1*, 1542–1552.
44. Dasgupta, A.; Kumar, R.; Sarlós, T. Fast Locality-Sensitive Hashing. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), San Diego, CA, USA, 21–24 August 2011*; ACM: New York, NY, USA, 2011; pp. 1073–1081. <https://doi.org/10.1145/2020408.2020578>.

45. Jia, Y.; Wang, J.; Zeng, G.; Zha, H.; Hua, X.S. Optimizing KD-Trees for Scalable Visual Descriptor Indexing. In *Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), San Francisco, CA, USA, 13–18 June 2010*; IEEE: Piscataway, NJ, USA, 2010; pp. 3392–3399. <https://doi.org/10.1109/CVPR.2010.5540006>.
46. Aggarwal, C.C. *Data Streams: Models and Algorithms*; Advances in Database Systems; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2007; Volume 31. <https://doi.org/10.1007/978-0-387-47534-9>.
47. Bifet, A.; Gavaldà, R.; Holmes, G.; Pfahringer, B. *Machine Learning for Data Streams: With Practical Examples in MOA*; MIT Press: Cambridge, MA, USA, 2018. <https://doi.org/10.7551/mitpress/10654.001.0001>.
48. Guirguis, S.; Sharaf, M.A.; Chrysanthis, P.K.; Labrinidis, A. Optimized Processing of Multiple Aggregate Continuous Queries. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM '11), Glasgow, UK, 24–28 October 2011*; ACM: New York, NY, USA, 2011; pp. 1515–1524. <https://doi.org/10.1145/2063576.2063793>.
49. Krawczak, M.; Szkatuła, G. An approach to dimensionality reduction in time series. *Inf. Sci.* **2014**, *260*, 15–36.
50. EarthScope Consortium. *NSF Seismological Facility for the Advancement of Geoscience (SAGE)*; EarthScope Consortium: Washington, DC, USA, 2026. Available online: <https://service.earthscope.org/> (accessed on 8 February 2026).
51. De Paepe, D.; Nieves Avendano, D.; Van Hoecke, S. Implications of Z-Normalization in the Matrix Profile. In *Pattern Recognition Applications and Methods*; De Marsico, M., Sanniti di Baja, G., Fred, A., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2020; Volume 11996, pp. 95–118. https://doi.org/10.1007/978-3-030-40014-9_5.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.