2023 20th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA) | 979-8-3503-1943-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/AICCSA59173.2023.10479252

TiVEx: Optimized Processing for Time Series Visual Exploration

Heba Helal*†, Mohamed A. Sharaf*, Mohammad M. Masud*, Panos K. Chrysanthis‡
*College of Information Technology, United Arab Emirates University, UAE
Email: {202090034, msharaf, m.masud}@uaeu.ac.ae
‡School of Computing and Information, University of Pittsburgh, United States of America
Email: {panos}@cs.pitt.edu

***Abstract*—To facilitate fast-visual data analysis, there is a need for recommending top-k views with "interesting" insights automatically. However, working with high-dimensional time series data makes the process of view recommendations difficult. The primary obstacle lies in finding an automatic way to generate views with less processing time (efficiency) while still closely aligning with the ground truth (effectiveness). In this paper, we propose TiVEx (Time Series Visual Exploration), a technique to address this challenge. TiVEx aims to achieve a balance between efficiency and effectiveness in generating view recommendations. Through extensive experiments, we demonstrate significant cost savings achieved by TiVEx, indicating its efficiency. Furthermore, our analysis delves into the exploration of striking the right balance between efficiency and effectiveness.**

***Index Terms*—Visualization, Recommendation, Time series data, Optimization**

I. Introduction

Time series data provides valuable information about various phenomena, ranging from economic indicators to environmental measurements [2], [3], [6], [8], [15], [18], [20]. However, visualizing and analysing multiple time series pose significant challenges due to the inherent complexities of high-dimensional data [8], [9], [15], [18].

To gain valuable insights from time series data, it is common for analysts to go through a visual data exploration step, in which the aim is to discover interesting patterns. However, that task is typically performed manually, where analysts generate and examine numerous visualizations looking for interesting insights. Particularly, analysts must manually construct prohibitively number of queries, selecting various combinations of data subsequences from different time series. Further, they need to visually explore the results of those queries looking for insights, which is clearly an ad-hoc and labor-intensive process.

While multiple research efforts have focused on automated visual data exploration systems (e.g., [5], [7], [11], [14], [17]), they fall short in handling the high-dimensionality nature of data. Particularly, the main idea underlying those systems is to automatically generate all possible exploratory queries of the data, generate their corresponding visualizations, and recommend the top-k interesting ones. Meanwhile, the interestingness of a query is quantified using a utility metric applied to its result. Hence, at each point during a data exploration session, automated visualization recommender systems need to search and process a large space of possible exploratory queries in order to recommend those visualizations with insightful patterns. That space of possible queries is further escalated when dealing with a large number of high-dimensional, long times series, where each timestamp in the series represents a dimension in a high-dimensional data space. Hence, there is a need for novel optimized query processing techniques that are specially tailored for exploring high-dimensional time series data. These techniques should quickly process and recommend interesting visualizations, which is the focus of this work.

In particular, the primary focus of our paper is to address the challenges of automatically generating and recommending top-k visualizations with interesting insights from time series data. To achieve this, we propose an efficient and effective method, called TiVEx (**Ti**me Series **V**isual **Ex**ploration). In TiVEx, similar to prior work (e.g., [7], [14], [17]), we adopt a *deviation-based* formulation to quantify the interestingness of an insight. As such, TiVEx is particularly designed for the data exploration tasks, in which the analyst wants to discover pairs of time series subsequences that exhibit a high level of dissimilarity. Hence, under TiVEx, each recommended visualization (i.e., view) is a pair of time series subsequence with a high Euclidean distance. The larger distances are often deemed more interesting as they highlight distinctive patterns or unusual occurrences that can lead to valuable insights [7], [14], [17].

To expedite the view recommendation process in our proposed method, we employ the pane window technique based on the notion of sharing of computation [13]. This technique takes advantage of the overlapping computations that occur when evaluating pairwise subsequences. Instead of computing each view individually and from scratch, we optimize the process by reusing calculations that have already been performed. This approach significantly reduces the processing time required to generate and evaluate the views. Our method achieves a substantial improvement in efficiency (i.e., less processing time) without compromising the quality of the recommended views (i.e., effectiveness).

The remaining sections of this paper are structured as

†Dept. of Computer Engineering and Systems, College of Engineering, Mansoura University, Egypt

follows: Section II describes the view recommendation process for time series data. Section III analyzes the challenges of recommending visualizations for time series data. Section IV presents TiVEx, an optimization technique designed to speed up the extraction of views with interesting insights. Section V describes extensive experiments and evaluation using the Google mobility dataset, and the paper is concluded with directions to future works in Section VI.

II. Preliminaries

A. Time Series Data

In the following, we present a formal definition of a time series, which is then extended to define multiple time series [3], [15], [18].

Definition 1: **Time series T_j**: A time series T_j is defined as an ordered set of observations $x_{i,j}$ and the corresponding timestamps $t_{i,j}$ at which each observation is recorded. Formally, $T_j = \{(x_{1,j}, t_{1,j}), (x_{2,j}, t_{2,j}), .., (x_{L,j}, t_{L,j})\}$, where L is the length of time series.

Definition 2: **Multiple time series T**: A multiple time series T is the set of n individual time series: $T = \{T_1, T_2, .., T_n\}$.

For a given time series T_j, the segment between any two data points $x_{b,j}$ and $x_{e,j}$ is called *subsequence* and is denoted as $s^j_{b,e}$. The subsequence $s^j_{b,e}$ starts at position b and ends at position e relative to the entire time series [18].

Example. To illustrate the basic concepts discussed above, we present an example based on the Google mobility dataset [1]. Since we also utilize that dataset for our experimental evaluation (see Section V), in the following we provide a brief overview of the Google mobility data. Particularly, Google's data records changes in mobility, quantified as mobility change ratio, across six different categories of places (i.e., Residential, Workplace, etc.). For each category, this mobility change ratio is calculated by tracking the visitor counts across different regions on a daily basis. These visitor counts are subsequently compared to a baseline day, where this baseline day corresponds to the median value of the 5 weeks spanning from January 3^{rd} to February 6^{th}, 2020.

Table I shows a sample of the Google mobility dataset for the UAE. There are two time series (i.e., T_1: Workplace and T_2: Residential), each one with four observations corresponding to four dates. For example, the four observations $(x_{1,1}, x_{2,1}, x_{3,1}, x_{4,1})$ for Workplace series T_1 are -46, -47, -44, and -37 corresponding to the timestamps April 1^{st}, 2^{nd}, 3^{rd}, and 4^{th} $(t_{1,1}, t_{2,1}, t_{3,1}, t_{4,1})$, respectively. In addition, the negative values in Workplace indicate how the number of visitors is lower than its baseline on that day. In contrast, the positive values in Residential correspondingly indicate the increase compared to the baseline of that day.

TABLE I: Sample of Google mobility dataset for UAE.

Date	Workplace (T_1)	Residential (T_2)
01/04/2020	-46	25
02/04/2020	-47	28
03/04/2020	-44	22
04/04/2020	-37	21

B. View Recommendation

Our model for visual data exploration relies on a database denoted as D_B, which stores multiple time series, each of length L. Each of these time series consists of an ordered list of real values representing observations [18]. The process of visual data exploration is initiated when the analyst submits query Q on D_B [7], [14]. Hence, the formal query Q is constructed as follows:

$$Q: \text{ SELECT } * \text{ FROM } D_B;$$

The result from query Q is a set of all the time series available in the database D_B, such as Workplace (T_1) and Residential (T_2) shown in Table I. The primary objective is to find interesting insights from the result of query Q. Specifically, we focus on pairs of time series, each with length L.

In order to extract insightful patterns from these pairs of time series, all possible pairs of aligned subsequences are extracted. In our scheme, each of these pairwise subsequences represents a particular visualization, denoted as a "view". Thus, the generation of all potential views, denoted as v, involves traversing through the pairs of lengthy and equal-length time series to obtain each possible pairwise subsequences. These generated pairwise subsequences are with arbitrary length, leading to a vast number of generated paired subsequences (i.e., views).

The manual process of generating views and identifying the views with interesting insights is considered a time consuming and labour-intensive task [5], [11], [17]. Therefore, our visual time series data exploration addresses this challenge by automatically generating and recommending top-k views with interesting insights.

In the context of time series data exploration, evaluating the interestingness of the generated views is a crucial step in the automatic process of view recommendation. Thus, to assess the generated views, we utilize the concept of distance between data points in two subsequences [5], [7], [11], [17]. Specifically, we employ the widely-used Euclidean distance metric, which quantifies the similarity or dissimilarity between subsequences of equal length. The Euclidean distance calculates the geometric distance between corresponding data points in the pairwise subsequences, providing a quantitative measure of (dis)similarity.

Considering the substantial number of pairwise subsequences produced from pairs of time series, our approach assumes a specific subsequence length R, which is user-defined parameter. By defining the subsequence length using the start and end positions of the subsequence (e.g., $s^j_{b,e}$) as $R = e - b + 1$, we are able to control the length of subsequences and, in turn, the number of views generated. In this context, the start b and end points e for each view, represented by aligned paired subsequence (i.e., $s^j_{b,e}$ and $s^k_{b,e}$), determine the position of the pairwise subsequences relative to the entire time series. Thus, a particular view $v_{s^j_{b,e}, s^k_{b,e}} \in v$ over D_B is represented by the tuple $< T_j, T_k, b, e >$. By determining the start and

end points for forming the subsequence, a particular view is generated from the user-specified pair of time series (i.e., T_j and T_k).

To ensure that all views are scaled consistently, we apply min-max normalization to the data points of each subsequence individually (e.g., $s^j_{b,e}$ or $s^k_{b,e}$) [10]. For instance, after normalization, the data point $x_{i,j}$ in the subsequence $s^j_{b,e}$ will be $X_{i,j}$, such that $X_{i,j} = \frac{x_{i,j}-\min}{\max-\min}$, $\max = \max^e_{i=b}(x_{i,j})$, and $\min = \min^e_{i=b}(x_{i,j})$. In this case, the values of data points in each subsequence (e.g., $s^j_{b,e}$ and $s^k_{b,e}$) range from 0 to 1. Hence, for the two subsequences $s^j_{b,e}$ and $s^k_{b,e}$ extracted from the two series T_j and T_k, the Euclidean distance for a certain view $v_{s^j_{b,e},s^k_{b,e}}$ is calculated using Eq. (1).

$$D(v_{s^j_{b,e},s^k_{b,e}}) = \sqrt{\sum_{i=b}^{e}(X_{i,j} - X_{i,k})^2} \quad (1)$$

where $X_{i,j}$ and $X_{i,k}$ are the data point after normalization for two subsequences $s^j_{b,e}$ and $s^k_{b,e}$, respectively. The positions b and e identify the start and end of a specific view, respectively.

By computing the Euclidean distance for each view, we obtain a numerical value that represents the deviation of the paired subsequences. Based on the assumptions that views with larger distances may reveal more intriguing insights [17], we focus on identifying the top-k views with the largest deviations. These top-k views are expected to highlight patterns, or interesting insights within the time series data [7], [17].

Despite our previous restriction on subsequence length, the search space for the extracted views remains substantial. The number of possible views is given by $V = L - R + 1$, which becomes inefficient for long time series, especially when the subsequence starts at any timestamp. Therefore, an optimized view recommendation technique is crucial for efficiently obtaining interesting insights with minimum processing time. Further details on the optimization technique will be discussed in Section IV-B.

III. Problem Definition

In a nutshell, our primary goal is to recommend the top-k most interesting visualizations based on the user's input query Q over a time series database D_B. Particularly, the process of extracting and recommending such views aims to uncover insightful patterns within the data using a specific scoring function. In this work, we focus on insightful patterns that are based on the (dis)similarity between pairs of aligned subsequences. However, the large number of possible pairs of subsequences results in a huge search space, which poses a computational challenge. To limit that search space, we assume a constraint on the subsequence length, such that we only search for pairwise subsequences of length R, which is a user-defined parameter. Despite that restriction on subsequence length, the search space remains vast as a subsequence of length R can start at any arbitrary position along the time series. Consequently, to further tame the search space, we employ a "sliding window" approach for time series exploration.

To employ a sliding approach for data exploration, there is a need for introducing a shift length parameter, which we denote as S. Given the subsequence length R and shift length S, a time series T is simply partitioned into overlapping subsequences, where a new subsequence starts every S time units and is of length R time units, as depicted in Fig. 1. Hence, the total number of different subsequences obtained from a time series of length L is V, such that $V = \frac{L-R}{S} + 1$. Particularly, each of those subsequences obtained from a time series T_i is expressed as $s^i_{(y-1)S+1,(y-1)S+R}$. That is, it starts at position $(y-1)S+1$ and ends at position $(y-1)S+R$, where $1 \leq y \leq V$. In turn, the set of possible views v over a pair of time series T_j and T_k is basically the set of overlapping pairwise subsequences of length R, such that each of which begins every S shift length. For each possible view (e.g., $v_{s^j_{b,e},s^k_{b,e}}$), the Euclidean distance metric is calculated using Eq. (1). Such metric quantifies the (dis)similarity between pairs of subsequences and serves as a measure of interestingness. Finally, we retrieve the top-k views with the largest distances, which are expected to contain the most interesting insights within the time series data.

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

Definition 3: **View Recommendation for Time Series Data:** Given two time series (T_j, T_k), a subsequence length R, a shift length S and a positive integer k, the goal is to find the top-k views with the highest utility score, where each view is an aligned pair of subsequences of length R.

IV. The TiVEx Approach for View Recommendation

In this section, we first discuss a linear search scheme for view recommendation, which serves as our baseline method. Subsequently, we present our TiVEx scheme, which leverages shared computation towards optimizing the process of view recommendation.

A. Baseline Solution

In linear search, all possible views of paired subsequences are generated. Consequently, the utility score (i.e., Euclidean distance) is computed for each of those views, and the top-k views with the highest scores will be recommended to the user. Clearly, linear search is a rudimentary and brute-force approach. As such, in this work, we use linear search as a baseline to assess the performance of our proposed TiVEx, which is described in the next subsection.

To further understand the performance of linear search, consider Fig. 1, which depicts two time series T_j and T_k of length $L = 20$ each. In this example, we assume that users are interested in subsequences of length $R = 8$, and that a slide distance $S = 6$ is used to enable a sliding window exploration of the two time series. Consequently, three possible views can be generated (i.e., $|V| = 3$). For instance, the two subsequences $s^j_{1,8}$ and $s^k_{1,8}$ are used to create the first view. In this setting, linear search will have to calculate the Euclidean distance for each of the three views before obtaining the views with the top-k highest distances. The Euclidean distance is

calculated for each view as in Eq. (1), where the first pairwise subsequence is located at arbitrarily position (b) relative to the entire time series. For instance, in Fig. 1, the start (b) and end (e) positions for both $s^j_{1,8}$ and $s^k_{1,8}$ are 1 and 8, respectively.

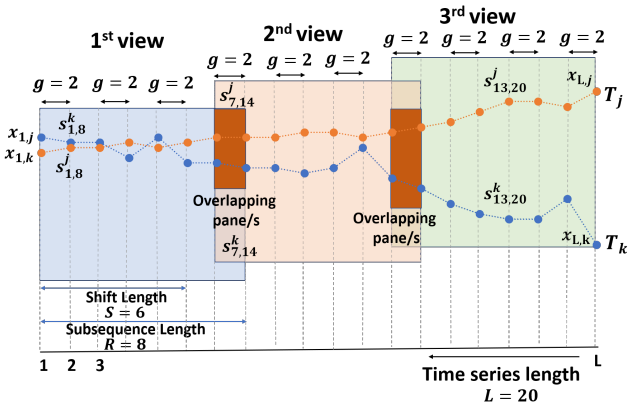


Fig. 1: The subsequences and views obtained from two time series T_j and T_k with $L = 20$, $R = 8$ and $S = 6$.

In order to assess the efficiency of the linear search method presented above, it is necessary to compute its total cost in terms of number of computations. Particularly, in this work, the overall cost is computed as the sum of all the operations needed to calculate the Euclidean distance for each paired subsequence, which is presented in Eq. (2) below:

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

Particularly, the cost (i.e., number of operations) for measuring the Euclidean distance within a pairwise subsequence can be calculated in terms of two components as follows:

1) Distance operations: The operations required to compute each of the terms $(X_{i,j} - X_{i,k})^2$ in Eq. (1), which are calculated R times for a subsequence of length R points.
2) Aggregate operations: The operations required to compute the overall summation $\sum_{i=b}^{b+R-1}(X_{i,j} - X_{i,k})^2$, which is calculated $(R-1)$ times for a subsequence of length R points, assuming that the subsequence starts at an arbitrary point b.

Combining those two components results in incurring a total cost of $R + (R-1)$ operations. Hence, for a total of $|V|$ pairs of subsequences (i.e., views), the overall cost would be $(2 \times R - 1) \times V$, as shown in Eq. (2). Notice, that in line with previous research [4], [10], [15], we omit the square root operation from our calculations since the Euclidean distance is monotonic. Clearly, in linear search the distance computation is performed individually for each view from scratch. In order to increase the efficiency of the exploration process, an optimization strategy is needed, which will be explained in the following subsection.

B. The TiVEx Scheme

In this subsection, we will explain the TiVEx technique, a scheme that operates on the concept of computation sharing through the use of pane window. In addition, we will shed light on how the exploitation of overlapping points plays a crucial role in influencing the processing time.

Pane window approach, which is used for query aggregation in streaming data, was first described in the paper [13]. The aim is to reduce the query execution time and storage. It entails separating the streaming data into multiple equal-sized, non-overlapping panes. In addition, each pane is handled independently before the outcomes are aggregated.

This research expands upon the concept of pane window within the context of time series, introducing a technique known as "TiVEx". The concept of pane window plays a crucial role in optimizing the process of view recommendation, enhancing the efficiency of exploratory data. In this technique, TiVEx, every subsequence of length R in the time series is divided into slices of equal size referred to panes. The determination of the pane size, denoted as g, is a key factor and is based on the combination of the subsequence length R and the shift length S. This is achieved by calculating the greatest common divisor (gcd) of R and S, yielding the pane size $g = \gcd(R, S)$. Using pane window in "TiVEx" technique helps in minimizing the redundancies and enhancing the overall efficiency of the view recommendation process of time series data, as we will explore further in the following discussions.

Applying TiVEx to a time series is demonstrated in Fig. 1. TiVEx technique divides each subsequence into panes with equal size $g = \gcd(8, 6) = 2$. Therefore, TiVEx utilizes the overlapping pane/s existing between any two successive subsequences (e.g., $S^j_{1,8}$ and $S^j_{7,14}$). The overlapping panes are shown in the shaded area in the middle of Fig. 1 (e.g., points 7 and 8).

Similar to [12], TiVEx technique divides a subsequence of length R into $\frac{R}{g}$ panes of equal size (e.g., 4 panes in 1^{st} view in Fig. 1). This indicates that each subsequence is split into partial aggregates representing each pane of size g. Consequently, the Euclidean distance calculation for a certain pairwise subsequence, starting at arbitrary position b, is performed as defined in Eq. (3). The calculation is carried out for each pane/partial aggregate individually (i.e., $S(g)$), as shown in the inner summation of Eq. (3). These individual pane distances are further assembled to form the final aggregation ($F() = \sum S(g)$), as depicted in the outer summation of Eq. (3).

$$D(v_{s^j_{b,e}, s^k_{b,e}}) = \sum_{c\,=\,1}^{R/g} \sum_{i\,=\,b\,+\,g\,\times\,(c-1)}^{b\,+\,c\,\times\,g\,-\,1} (X_{i,j} - X_{i,k})^2 \tag{3}$$

In order to compute the overall cost for measuring the Euclidean distance along all partial aggregates, the following three items are calculated:

1) Distance operations: The operations required to compute $(X_{i,j} - X_{i,k})^2$ for all points inside the partial aggregate/-pane, with g points for each pane.
2) Aggregate operations: The operations required to compute the summation $\sum_{i=b}^{b+g-1}(X_{i,j} - X_{i,k})^2$ inside the partial aggregate, which is calculated $(g-1)$ times for

a pane of length g points. The subsequence is assumed to start at an arbitrary point b.

3) Final aggregate: The operations required to calculate the final summation between the partial aggregates. It is calculated ($\frac{R}{g} - 1$) times for a subsequence with $\frac{R}{g}$ number of panes.

The first two items are applied within a specific partial aggregate/pane; however, the last one is applied along the partial aggregates. In Fig. 1, the first pane in $S^j_{1,8}$ and $S^k_{1,8}$, starting at point $b = 1$ and with size $g = 2$, is considered the first partial aggregate which applies distance operation (i.e., $(X_{i,j} - X_{i,k})^2$) between the first two points in $S^j_{1,8}$ and the corresponding two points in $S^k_{1,8}$ and then applies aggregate operation (i.e., $\sum_{i=1}^{2}(X_{i,j} - X_{i,k})^2$) to sum the two previous distance operations. The previous calculation of partial aggregate is applied to all available panes existing in $S^j_{1,8}$ and $S^k_{1,8}$ (i.e., four panes). In the final aggregation step, the outcomes from the different partial aggregates are combined to form the final result.

Moreover, there are partial aggregates (also called sliced aggregates) representing the overlapping aggregates between two successive subsequences, such as the pane with size $g = 2$ in Fig. 1 existing from point 7 to point 8. This overlapping/sliced aggregate is buffered to be used directly in the upcoming subsequences (i.e., $S^j_{7,14}$ and $S^k_{7,14}$, respectively). In this instance, the total cost of all obtained views is as follows:

$$\begin{aligned} Total\ Cost &= (2 \times R - 1) \\ &+ (2 \times S - 1 + \frac{R - S}{g}) \times (V - 1) \end{aligned} \tag{4}$$

The cost calculation for the first view in TiVEx technique takes into account all available partial aggregates, which have a total length of R points. Therefore, the cost of the first paired subsequence is determined by the sum of the length of all partial aggregates, which is R points, plus the cost of combining them, which is $(R - 1)$. For example, the cost of the first view in Fig. 1 would be $8 + 7 = 15$.

In subsequent views (e.g., 2^{nd} and 3^{rd} views in Fig. 1), the overlapping aggregates are utilized to calculate the cost. It is important to note that in each successive subsequence, the remaining portion after excluding the overlapping aggregates has a fixed length of S points. Hence, the total cost of all partial aggregates in all views except the first one (denoted as $|V - 1|$) can be determined by considering the following:

1) The cost required to compute distance operations for all points inside the remaining partial aggregate/pane, with S points (e.g., from point 9 to 14 in the 2^{nd} view in Fig. 1). In this case, the cost is 6.
2) The cost required to compute the summation inside the partial aggregate, which is $(g - 1)$ within each of the $\frac{S}{g}$ panes. In the second and third views in Fig. 1, there are 3 panes each with a size of $g = 2$. Thus, the cost (e.g., 2^{nd} view) will be $\frac{6}{2}(2 - 1) = 3$.
3) The cost required to calculate the summation between these partial aggregates, which is calculated as ($\frac{S}{g} - 1$) between $\frac{S}{g}$ number of panes. In Fig. 1, 2 addition operations are required to assemble the 3 panes in the 2^{nd} and 3^{rd} views.
4) The cost required to combine the previous partial aggregates with each overlapping aggregates. There are $\frac{R-S}{g}$ overlapping partial aggregates (e.g., partial aggregate from point 7 to 8 in Fig. 1). Thus, only one addition operation is needed in the 2^{nd} and 3^{rd} views to combine the partial aggregates in the remaining portion with the one in the overlapping portion.

By considering these terms, the total cost is the same as in Eq. (4). This cost calculation allows for an efficient evaluation of the subsequent views in TiVEx, considering the overlapping nature of the partial aggregates.

V. Experimental Results and Analysis

In the experiments that follow, we aim to assess both the efficiency of TiVEx compared to linear search and the effectiveness of TiVEx under various parameter settings.

When designing and evaluating visualization systems for view recommendations, it is important to consider the trade-off between effectiveness and efficiency. Achieving highly effective recommendations often requires computationally expensive operations, which in turn can lead to longer processing times. Conversely, prioritizing efficiency may sacrifice some level of effectiveness by relying on simpler or less computationally intensive approaches. Therefore, the ultimate goal is to strike a balance between efficiency and effectiveness to ensure the optimal performance of visualization system.

To measure the effectiveness achieved by our proposed TiVEx scheme, we compare two lists: 1) the list of top-k recommended views when $S = 1$, which represents the ground truth obtained by exhaustive search, and 2) the list of top-k recommended views when $S > 1$, which represents an approximate solution that aims to reduce the search space. In general, a commonly used measure for evaluating the similarity between any two lists of ranked items (e.g., Z_1 and Z_2) is the Rank-biased Overlap (RBO) [19]. In the following, we first discuss the traditional RBO measure and subsequently elaborate on its adaptation to the context of our problem.

The RBO score ranges from 0 to 1, where 0 indicates no overlap between the two lists (i.e., dissimilar); however, 1 indicates a perfect overlap (i.e., identical) [19]. The RBO measure takes into account both the order (importance) of the items on the lists (e.g., item 1 is more important than item 2) and the overlap between the items in a certain order k (i.e., c_k). In addition, it considers two factors, depth (d) and p-value (p). The depth (d) determines the depth of the list up to which the similarity is measured, while p-value (p) controls the contribution of top-k items to the final value of RBO similarity measure, $0 < p < 1$. Higher values of p indicate a stronger preference for the top-k items. The RBO formula is as follows:

$$RBO(Z_1, Z_2, p, k) = \frac{c_k}{k} \times p^k + \frac{(1 - p)}{p} \times \sum_d \frac{c_d}{d} \times p^d \tag{5}$$

In this context, the items in the two lists represent the top-k recommended views with interesting patterns, where each view is defined by start and end positions relative to the entire time series (e.g., $v_{s^j_{b,e},s^k_{b,e}}$), as mentioned previously. Specifically, the views generated by our TiVEx technique represent pairwise subsequences when the shift length is $S > 1$. These views are compared with the corresponding pairwise subsequences generated when $S = 1$. Both ranked lists, Z_1 and Z_2, are of equal length, corresponding to the top-k views. However, comparing these two ranked lists solely using RBO measure poses a challenge. The RBO measure evaluates the similarity or dissimilarity between two views at a specific depth d, denoted as $c_d = 1$ for similarity and $c_d = 0$ for dissimilarity. In our case, where we compare pairwise subsequences (views) from different shift lengths (e.g., $v_{s^j_{b1,e1},s^k_{b1,e1}}$ and $v_{s^j_{b2,e2},s^k_{b2,e2}}$), we encounter various scenarios. These scenarios include complete similarity ($b_1 = b_2$ and $e_1 = e_2$), complete difference ($e_1 \leq b_2$), or partial overlap between the subsequences. To address these different scenarios, we enhance the RBO metric by incorporating the Jaccard similarity measure. The Jaccard similarity measure is commonly used to compare the similarity or dissimilarity between sets and considers the overlap between the two sets [16], making it convenient for our case. Thus, to calculate the intersection/overlap (c_d) between two pairwise subsequences for two different shift lengths (e.g., $v_{s^j_{b1,e1},s^k_{b1,e1}}$ and $v_{s^j_{b2,e2},s^k_{b2,e2}}$) at a certain depth d, the following formula is used:

$$\frac{e_1 - b_2 + 1}{e_2 - b_1 + 1} \tag{6}$$

Here, e_1 and e_2 denote the end positions, while b_1 and b_2 represent the start position of the views, and $b_1 \leq b_2$. The case where $e_1 \leq b_2$, the intersection (c_d) is 0 (i.e., no overlap). When $b_1 = b_2$ and $e_1 = e_2$, the intersection (c_d) is 1 (i.e., similar). By comparing the positions of the pairwise subsequences, we can quantify the overlap and determine the contribution of the pairwise subsequences (views) at depth d. This enhanced RBO measure provides an effective means of evaluating the quality of the resulting view recommendations, enabling assessment of the similarity between the generated views at $S > 1$ and the views obtained at $S = 1$ (i.e., ground truth).

Prior to evaluating TiVEx's efficiency and comparing it to linear search, we demonstrate how the view recommendation uses the Euclidean distance to help offer insightful information from the investigated dataset. We took a portion of the Google Mobility dataset for the UAE. The dataset covered the period from February 15, 2020, to May 24, 2020, and had a length of $L = 100$, as in Fig. 2. By applying a sliding approach with a subsequence length $R = 20$ and a shift length $S = 8$, a total of 11 views are extracted from the dataset. Each view corresponds to a pairwise subsequence, and a key feature of this derivation is the overlapping nature of these pairwise subsequences. This underlying characteristic notably enriches the subsequent analytical process.

The process of automated view recommendation is executed as follows: First, each pairwise subsequence undergoes normalization, followed by the application of TiVEx. This involves the computation of the Euclidean distance for each view, incorporating considerations for the savings through the utilization of overlapping panes between the consecutive views. Upon analyzing the results, the top-2 views exhibiting the highest distances are identified as view 4 and view 3, as shown in Fig. 2 (a) and (b), respectively. These two views are promptly and automatically recommended to the analyst. The insights offered by these views unveil interesting patterns related to the behavior of people during the COVID-19 pandemic and the subsequent lockdown. In the early stages of the pandemic, the workplace indicator exceeds the residential indicator. However, after the lockdown, there is a notable shift, with people staying at home rather than going to work. This observation highlights the impact of the lockdown on mobility patterns.

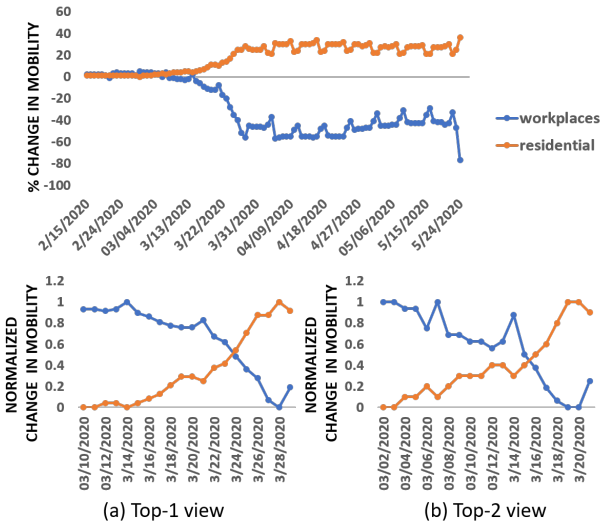
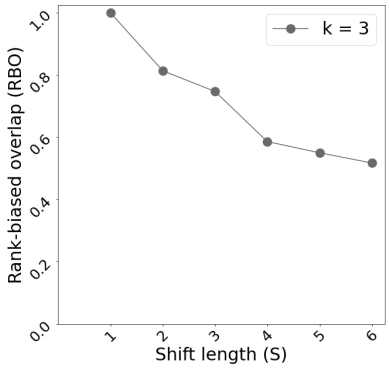


Fig. 2: View recommendation for a dataset of length $L = 100$, subsequence length $R = 20$ and shift length $S = 8$.

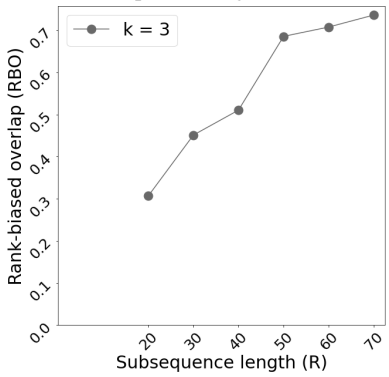
Fig. 3 (a) provides insightful observations into the effectiveness of different shift lengths by showcasing the value of updated RBO with a fixed value of $k = 3$. Across all shift lengths, there is a consistent trend of decreasing RBO values. This decrease can be attributed to the increasing deviation between the extracted views and the corresponding views with a shift length of $S = 1$. It is noteworthy that the shift length ($S = 1$) yields perfect results with RBO value of 1.

Furthermore, Fig. 3 (b) extends the analysis by examining the updated RBO measure for varying subsequence lengths, while keeping k fixed at 3. The results demonstrate that increasing the subsequence length leads to higher RBO values, as the deviation of the extracted views decreases in comparison to the baseline case with $S = 1$. For example, a subsequence length of $R = 70$ yields a higher RBO value, whereas a length of $R = 20$ results in the lowest RBO value.

In Fig. 4, we investigate the effect of different k values.



(a) Subsequence length $R = 20$.



(b) Shift length $S = 5$.

Fig. 3: Measuring effectiveness using updated RBO measure for $k = 3$, $L = 500$, and $p = 0.9$.

Consistent with the previous findings, we also observe a decreasing trend of RBO values across all shift lengths as k increases. This decline reflects the growing deviation between the extracted views and the corresponding views with $S = 1$. Despite the increase in distance at higher k values, the importance of the extracted views in the RBO calculation and their substantial differences from the ground truth case justify the decrease in RBO values for all shift lengths.

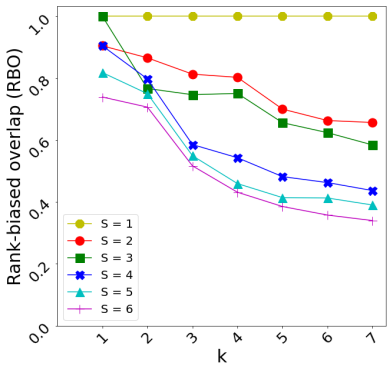
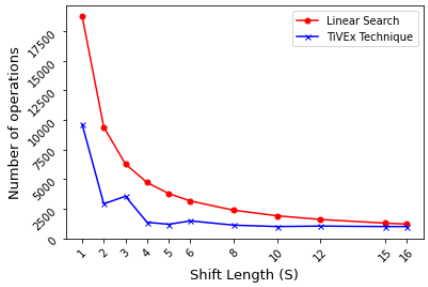
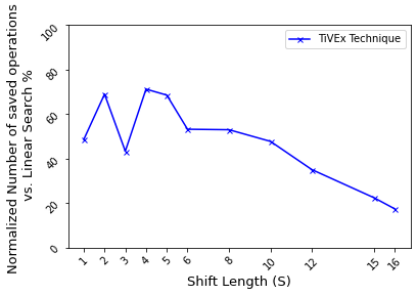


Fig. 4: Measuring effectiveness using updated RBO for different k, $L = 500$, $R = 20$, $p = 0.9$, and different shift lengths.

In this experiment, we evaluate the efficiency of TiVEx compared to linear search. Efficiency is assessed by measuring the number of operations required and the resulting savings for both techniques. The cost comparison for different shift



(a)



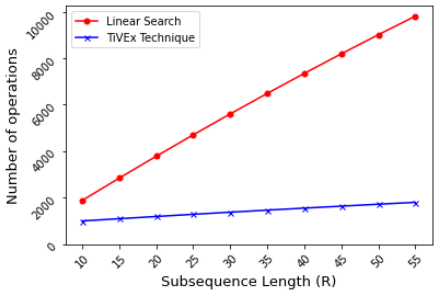
(b)

Fig. 5: Comparing the cost of linear search and TiVEx for a dataset of length $L = 500$, $R = 20$, and varying S.

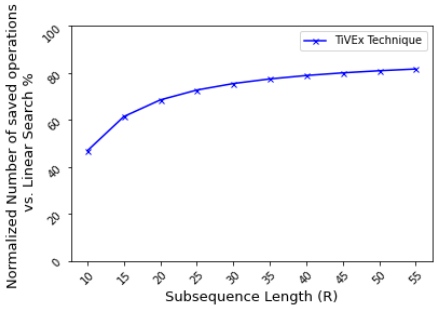
lengths S is illustrated in Fig. 5, with a focus on shift lengths that yield an integer number of views to ensure a fair comparison. The results shown in Fig. 5(a) clearly indicate that linear search requires significantly more operations compared to TiVEx. As the shift length increases, the number of operations decreases for both linear search and TiVEx. This decrease can be attributed to two factors: 1) as the shift length increases, the number of generated views decreases, leading to a reduction in the number of operations; and 2) TiVEx experiences a decrease in the number of overlapping points between successive subsequences as the shift length increases, further contributing to the decrease in operations (i.e., combine partial aggregates in the remaining part with each overlapping aggregate).

However, the combination of subsequence length (R) and shift length (S) affects the pane size, which in turn impacts the total number of operations required. When the R and S combination results in a small pane size, the number of operations required to assemble the panes increases, leading to slight irregularities observed in the cost of TiVEx (e.g., at $S = 3$ and $S = 6$). Despite the irregularities, TiVEx still demonstrates a significant advantage over linear search in terms of cost. Although the difference in cost between the two techniques decreases as the shift length becomes larger, there is still a substantial disparity, with TiVEx showcasing a 17% advantage over linear search at $S = 16$.

Fig. 5(b) presents valuable insights on the normalized savings achieved by TiVEx, as compared to the number of operations required in linear search, considering different shift length values S. The figure reveals that TiVEx provides substantial savings for small S. However, as S increases, the



(a)



(b)

Fig. 6: Comparing the cost of linear search and TiVEx for a dataset of length $L = 500$, $S = 5$, and varying R.

savings decrease, indicating a reduction in the number of overlapping points between successive subsequences. These findings from Fig. 5 align with the analysis presented in Section IV, confirming the advantages of TiVEx.

Similarly, the experiments are conducted considering different subsequence length values, and the results are shown in Fig. 6. Fig. 6 (a) demonstrates that linear search outperforms TiVEx in terms of the number of operations required. Both linear search and TiVEx exhibit an increasing trend as the subsequence length grows. When the subsequence length increases, each pairwise comparison between subsequences becomes more computationally intensive. As the subsequence length grows, the number of data points to compare within each subsequence also increases, resulting in a larger number of operations required for comparisons.

Fig. 6 (b) provides insights into the normalized savings achieved by TiVEx when compared to the operations required in linear search. The figure demonstrates that the normalized savings increase as the subsequence length R grows, indicating a higher number of overlapping points. This finding highlights the efficiency of TiVEx in terms of saved operations, particularly in scenarios with longer subsequence lengths and a larger degree of overlap.

A careful examination of Fig. 3, 5(b), and 6(b) reveals an important trend: using smaller shift lengths and larger subsequence lengths leads to higher savings in terms of operations and higher RBO values. This observation leads us to the conclusion that striking a balance between efficiency and effectiveness in the view recommendation process is achieved by employing a small shift length and a large subsequence length. These parameter settings allow for efficient computation while still producing recommendations that closely align with the ground truth.

VI. Conclusions

In the realm of data analysis and exploration, obtaining the most interesting visualizations is a crucial objective, yet it often involves a laborious and time-consuming process. To address this challenge, we propose an optimized technique named TiVEx, which leverages computation sharing to enable efficient visual analysis. By reducing the computational time required for view recommendations, TiVEx streamlines the exploration process. We also conduct a thorough investigation into achieving the optimal trade-off between efficiency and effectiveness in generating insightful visualizations. As future work, we plan to explore additional optimization techniques, such as pruning, to further enhance processing time.

VII. Acknowledgments

This research is supported partially by UAE University Strategic Research Program Grant (12R147). Additionally, we thank Nishi Kochunni for her valuable feedback on this work.

References

[1] https://www.google.com/covid19/mobility, accessed: 2023-05-14.
[2] Aghabozorgi, S., Shirkhorshidi, A. S., and Wah, T. Y. "Time-series clustering–a decade review." Information systems 53 (2015): 16-38.
[3] Benkabou, S. E., et al. "Unsupervised outlier detection for time series by entropy and dynamic time warping." KAIS 54.2 (2018): 463-486.
[4] Buono, P., et al. "Interactive pattern search in time series." Visualization and Data Analysis 5669 (2005): 175-186.
[5] Ding, R., et al. "Quickinsights: Quick and automatic discovery of insights from multi-dimensional data." MOD (2019).
[6] Sharaf, M. A., et al. "CovidLens: Visually Understanding the Covid-19 Indicators through the Lens of Mobility Data." MDM (2022): 302-305.
[7] Ehsan, H., Sharaf, M. A., and Chrysanthis, P. K. "Efficient recommendation of aggregate data visualizations." TKDE, 30.2 (2017): 263-277.
[8] Gogolou, A., et al. "Comparing similarity perception in time series visualizations." IEEE transactions on visualization and computer graphics 25.1 (2018): 523-533.
[9] Keogh, E., et al. "Finding the most unusual time series subsequence: algorithms and applications." Knowledge and Information Systems 11 (KAIS) (2007): 1-27.
[10] Keogh, E., and Kasetty, S. "On the need for time series data mining benchmarks: a survey and empirical demonstration." ACM SIGKDD (2002).
[11] Key, A., et al. "Vizdeck: self-organizing dashboards for visual analytics." ACM SIGMOD (2012).
[12] Krishnamurthy, S., Wu, C., and Franklin, M. "On-the-fly sharing for streamed aggregation." ACM SIGMOD (2006).
[13] Li, J., et al. "No pane, no gain: efficient evaluation of sliding-window aggregates over data streams." ACM SIGMOD 34.1 (2005): 39-44.
[14] Sharaf, M. A., Mafrur, R., and Zuccon, G. "Efficient Diversification for Recommending Aggregate Data Visualizations." IEEE Access (2023).
[15] Rakthanmanon, T., et al. "Addressing big data time series: Mining trillions of time series subsequences under dynamic time warping." ACM TKDD 7.3 (2013): 1-31.
[16] Schütze, H., Manning, C. D., and Raghavan, P. "Introduction to information retrieval." Cambridge University Press (2008).
[17] Vartak, M., et al. "Seedb: Efficient data-driven visualization recommendations to support visual analytics." VLDB 8.13 (2015): 2182.
[18] Wang, X., et al. "Experimental comparison of representation methods and distance measures for time series data." Data Mining and Knowledge Discovery 26 (2013): 275-309.
[19] Webber, W., Moffat, A., and Zobel, J. "A similarity measure for indefinite rankings." ACM TOIS 28.4 (2010): 1-38.
[20] Weber, M., Alexa, M., and Müller, W. "Visualizing time-series on spirals." Infovis 1 (2001).