2019 IEEE International Conference on Big Data (Big Data)

MTSAD: Multivariate Time Series Abnormality Detection and Visualization

Vung Pham
Computer Science Department
Texas Tech University
Lubbock, TX 79409, USA
vung.pham@ttu.edu

Ngan Nguyen
Computer Science Department
Texas Tech University
Lubbock, TX 79409, USA
ngan.v.t.nguyen@ttu.edu

Jie Li
Computer Science Department
Texas Tech University
Lubbock, TX 79409, USA
jie.li@ttu.edu

Jon Hass
Distributed Management Task Force
Dell, Inc.
Austin, TX, USA
jon.hass@dell.com

Yong Chen
Computer Science Department
Texas Tech University
Lubbock, TX 79409, USA
yong.chen@ttu.edu

Tommy Dang
Computer Science Department
Texas Tech University
Lubbock, TX 79409, USA
tommy.dang@ttu.edu

***Abstract*—Detecting outliers is one of the fundamental tasks in visual analytics and valuable in many application domains, such as suspicious network cyberattack recognition. This paper introduces an approach to analyzing and visualizing high-dimensional time series, focusing on identifying multivariate observations that are significantly different from the others. We also propose a prototype, called MTSAD, to guide users when interactively exploring abnormalities in large time series. The prototype contains two views: the main window provides an overview of identified outliers overtime, the detail window investigates and explores the ranked temporal data entries based on their outlying contributions to the overall plots. The visual interface supports a full range of interactions, such as lensing, brushing and linking, ranking, and filtering. To validate the benefits and usefulness of our approach, we demonstrate MTSAD on real-world datasets of different numbers of attributes.**

***Index Terms*—Multivariate Time Series Visualization, Outlier vs. Inlier, Abnormality Detection, Box Plot Rule, Radar Charts, Parallel Coordinates.**

I. Introduction

Multivariate time series data is growing in terms of both the number of time steps and the number of dimensions from various application domains [1], such as cybersecurity, business, and health care. For example, each state in the United States can be considered as a data entry that is measured based on various economic sectors (such as manufacturing, business, education, and farming). These economic indicators change every month. An example outlier in a given month is a state which has significant drops on these economic indicators due to hurricane [2].

Abnormality detection is one of the critical applications of this type of data. An abnormal point is one that is significantly different from the others in a dataset. In multivariate data, an unusual point can be identified by a significant difference in individual dimension or in a combination [3]. It may indicate adverse effects such as fraud and security breach or merely noises. It may also have positive indications, such as abnormally high website traffic due to releases of a new product. In either case, these unusual behaviors are often of interest to organizations [4].

Abnormality detection with multiple outliers becomes more complicated due to the masking and swamping effects [5]. The masking effect is that an outlier is not detected due to the presence of another outlying point. On the other hand, the swamping effect is that a regular observation is wrongly identified as an outlier because of the presence of another point(s) [6]. Highlighting masking and swamping effects are challenging and computationally expensive. In this paper, we tackle this problem by introducing a leave-one-out mechanism for high-dimensional data. The idea is straightforward: we leave a data entry out and recompute the outlying score of the new plot and compare it to the original one [7].

Our contributions in this paper thus are:

- We propose an interactive prototype, MTSAD, to highlight and explore outliers in high dimensional time series. The visual interface supports a full range of interactive operations, such as lensing, brushing and linking, ranking, and filtering.
- We demonstrate the benefits of our approach by using MTSAD on real-world datasets. The proposed technique can be scaled and applied to other data arisen in various application domains.

The paper is structured as follows: We describe related work in the following section. Then we introduce our MTSAD components and implementation through visual examples. We present MTSAD application on multivariate health metric data of 467-node cluster at the High-Performance Computing Center at Texas Tech University in the Use Cases section. We argue that by investigating higher dimensional data, we can detect the outliers which are usually not discernable in lower dimensions. Finally, we conclude our work and present future research direction.

978-1-7281-0858-2/19/$31.00 ©2019 IEEE

II. Related work

Time series data contains the values of monitoring variables collected over time. A time series (X) can be formally defined as: $X = \{x^1, x^2, ..., x^n\}$, where n is the number of observed time steps and $x^t = \{x_1^t, x_2^t, ..., x_m^t\} \in R^m$ is the observed values for m monitoring variables at time t (from 1 to n). The observations of a single variable ($m = 1$) over time make univariate time series. On the other hand, the observations of multiple variables ($m > 1$) over time make multivariate time series.

Multivariate time series are becoming more popular for organizations' monitoring tasks and so the need to detect abnormalities in the data over time. Therefore, there is a large number of researches in this field in the literature. The established and potential applications are ranging from cyber-security [8] and fraud detection [9] to environmental issues [10]. One of the familiar sources of this type of datasets is from large internet companies or high-performance computing centers. For these organizations, it is critical to monitor the server metrics (e.g., CPU utilization, memory utilization, and power usage), across time, and to detect unusual behaviors. Once identified, preemptive actions can be taken to provide better services (e.g., performances or securities). Examples of such monitoring systems are the cluster trace datasets from Google [11] and Alibaba [12].

A. Abnormality detection methods for multivariate time series

Abnormality detection methods are used to extract outlying information from data before visualizing them in our solution. There are many statistical methods applied to anomaly detection in the literature. Readers of interest can refer to a paper published by Wilkinson [3] on an excellent review of these techniques. One popular statistical outlier detection method is John Tukey's one [13]. This method defines Interquartile range (IQR) to detect outliers in a dataset. The IQR is defined as

$$IQR = 3^{rd}\ quartile - 1^{st}\ quartile \tag{1}$$

Then the region(s) contains the normal data points are within the upper and lower bounds, which are defined as:

$$upperbound = 3^{rd}\ quartile + IQR * factor \tag{2}$$

$$lowerbound = 1^{st}\ quartile - IQR * factor \tag{3}$$

where the $factor$ is usually set to 1.5, or the users could tweak this value to suit their applications and requirements.

Researchers also applied machine learning techniques to abnormality detection in large multivariate time series such as Auto-Regressive Integrated Moving Average (ARIMA) [14], and more recently, Support Vector Machines (SVM), tree-ensembles, and Artificial Neural Networks (ANNs) [15]. Specifically, Takeuchi and Yamanishi [16] proposed a unifying framework for detecting outliers and change points called *ChangeFinder*. They adopt ARIMA to learn a sequence of probability density functions denoted as $\{p^t : t = 1, 2, ...\}$ from the observed data. They then define a conditional probability density function to give the probability of values at time $t + 1$ (x^{t+1}) given the observed values before time t ($\{x^1, x^2, ..., x^t\}$) as $p^t(x^{t+1}|x^t, x^{t-1}, ..., x^1)$. These probability density functions can then be used to give outlying scores for the actual observed values at time $t+1$ (e.g., using logarithmic loss scores). Besides, these functions can also be used to predict the expected values at the time $t+1$ ($\hat{x}^{t+1}$). The outlying scoring metric can then be the quadratic difference between the predicted values and the actual observed data at time $t + 1$ (x^{t+1}).

On the other hand, SVM [17], specifically, One-class SVM had been applied to abnormality detection by Heller et al. [18] and gained initial success. This algorithm provides a binary function to identify the coverage region(s) for normal input data based on some probability density function defined by the users. This function returns +1 for a normal data point and -1 elsewhere. On the other hand, Isolation Forest (IF) as part of the tree-ensembles' techniques was also applied in abnormality detection by Liu et al. [19]. The main idea of IF is that decision trees would need a higher number of splits to separate a normal point from the others. On the contrary, it would require a significantly smaller number of splits to isolate an outlying point.

With the recent advancements of Artificial Neural Networks, they were also used in abnormality detection. In case of time series, it is natural to apply Long Short-Term Memory (LSTM) [20] ANNs in this problem. Malhotra et al. [21] implemented the idea with two main stages. First, at a given time point, the algorithm generates predictions for the future data using data observed before this point. It then compares the predicted values with the actual values to generate the outlying scores using dis/similarity metrics (e.g., root mean squared error).

These machine learning approaches require a large amount of data and a significant amount of time to train models before applying them for detecting abnormalities. These requirements are the barriers to using machine learning approaches in an online monitoring system in which the data changes dynamically and frequently. We aim to apply our solution in real-time monitoring tasks, which allow frequent changes in the environment. Therefore, in this work, we use Tukey's well-known statistical approach to outlier detection, which leverages the IQR as described above.

B. Multivariate time series visualization

We also incorporate visualization in our solution to envision the calculated outlying scores. Line-graphs and heat-maps are commonly used to visualize time series data [10]. Line-graphs can show trends and value differences well. However, it creates visual cluttering issues in application to datasets with a large number of time-series. On the other hand, heat-map helps to resolve the cluttering issue by leveraging the other visualization attribute (e.g., cell color). However, heat-maps require large display spaces for sizeable multivariate time-series. The use of animation can circumvent the need for displaying the entire dataset over time. However, it requires the analysts to memorize the visualization components in the previous several time steps or to replay the animation

several times while analyzing the data at a specific time. These requirements lead to less effectiveness for data analysis and data exploration [22].

Another approach to deal with the cluttering of visualization elements is to use small multiples [23]. Small multiples show snapshots of the data via a few consecutive time steps.

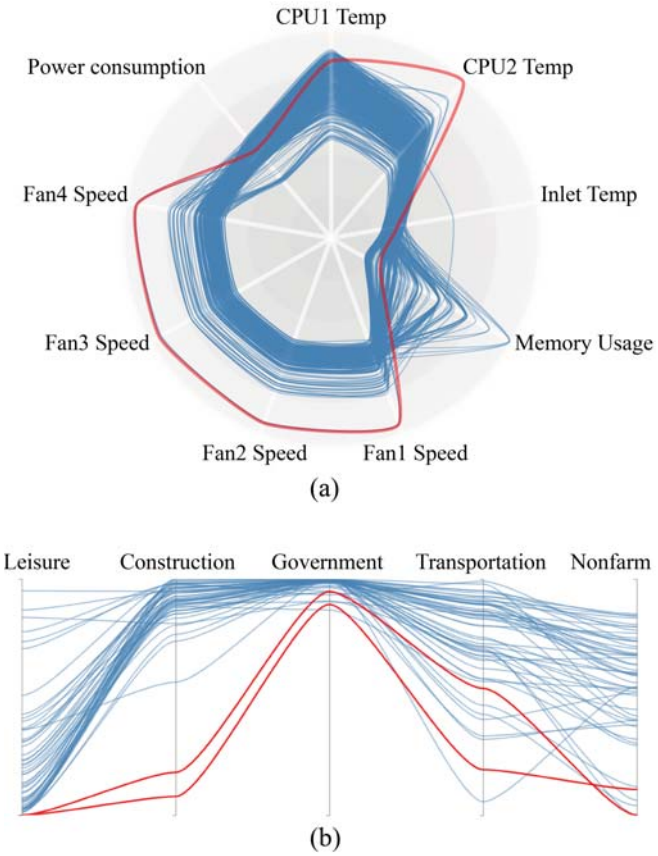

Fig. 1. Multivariate visualizations for different data: (a) Radar chart for 467 data entries of 9-dimensional computation nodes in a high-performance computing center (each path is a machine, and the red one is an outlier). (b) Parallel coordinates of the five economic sectors in the employment data retrieved from the U.S. Bureau of Labor Statistics (each curve is a state, and red curves are outliers).

For multivariate data visualization, radar chart [24] and parallel coordinates [25] are often used. Radar chart is a geometric projection to visualize multidimensional data in two-dimensional space in a circular manner. In a radar chart, equiangular axes are used to represent variables (which can be manually organized by users). The distance from the center (or a specified offset from the center) of the radar to the point on an axis represents the variable value [26]. A polygon connecting points representing variable values of a single item. This polygon characterizes the shape of that item. Figure 1 (a) gives an example of using a radar chart to visualize nine critical variables (i.e., *CPU1 Temp*, *CPU2 Temp*, *Inlet Temp*, *Memory Usage*, *Fan1 Speed*, *Fan2 Speed*, *Fan3 Speed*, *Fan4 Speed*, and *Power consumption*) for 467 computation nodes collected from a high performance computing center [27] at a university, at one time step. Each polygon path represents a computation node and the red path is the outlying one.

Similarly, the parallel coordinates [28] is another methodology for visualizing multivariate data based on a non-projective mapping between N-Dimensional and 2-Dimensional sets. The data items are represented by lines and hyperplanes, allowing the visualization of multidimensional relation rather than just finite point sets. Figure 1 (b) is an example of using parallel coordinates to visualize five economic sectors (i.e., *Leisure*, *Construction*, *Government*, *Transportation*, and *Nonfarm*) in the employment dataset retrieved from the U.S. Bureau of Labor Statistics [29]. The parallel coordinates are the five sectors, the paths represent the states, and the red paths are the outlying ones.

Our solution uses the radar chart instead of parallel coordinates since it can capture the morphology of multidimensional curves [30]. However, the choice of multivariate representations does not affect the underlying mechanism to compute outlying/inlying scores of the plots. In other words, the outliers/inliers should be visible in both radar charts and parallel coordinates (or other types of multivariate representations). Also, with the reasons discussed above, MTSAD uses small-multiples to visualize the time dimension of the multivariate time-series.

III. Proposed Solution

Our proposed solution to outlier detection is based on two main components: back-end outlying computation and front-end visualizations and interactions.

A. Outlier detection method

In multivariate data, different dimensions may have different ranges of values. So, the first step in our approach is to normalize the data for all variables into a specified range (e.g., [0, 1]). This step assures that no individual variable is going to dominate the outlying score in our computation. We aim to apply our solution to real-time monitoring tasks. In these, it is often required to provide abnormality detection in a real-time and unsupervised manner. Therefore, we use Tukey's well known outlying detection method utilizing IQR to specify the upper bound for the outlying score as described in (1) and (2). Our approach does not consider the lower bound as it is based on multidimensional distances of a point to the other points. If a point is far from the others based on the threshold produced by the Box-Plot rule, then it is outlying and is not otherwise. This approach also increases computation efficiency, since we would not need to consider distances from an individual point to all others. We only need to consider the minimum distance from that point to other data items. Meaning we are making use of the minimum spanning tree (MST) built from all data items in our outlying computation.

Figure 2 shows an example of MST for the stock data (*stock close value* vs. *stock volume*) at a time step from the New York Stock Exchange dataset collected from Kaggle [31]. On the left panel, each dot represents a stock. Also, on the right panel, green edges have normal MST lengths, while red edges

are significantly longer identified by Box-Plot rule. Therefore, red dots are considered as outlying stocks in our approach. Notice also that dots within a close proximity have been aggregated into bins [32] to simplify the MST and reduce the computational cost.

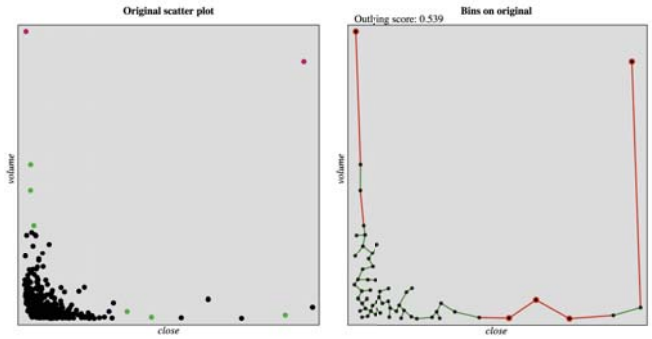

Fig. 2. An example scatterplot of *stock close value* vs. *stock volume* on the left and its MST on the right: red edges have lengths more than the threshold produced by the Box-Plot rule. Notice also that, on the right panel, the data points are aggregated into bins before computing the MST to reduce computation cost.

As part of the monitoring task, we are not only considering the outlying status of the whole set of items at once but also the outlying measure of every individual data point at a specified time step. To do this, MTSAD applies leave-one-out approach to outlying computation process. In this leave-one-out approach, the outlying score of the whole dataset is first computed, then every data item is left out, and the outlying score is recomputed. The difference between the two outlying scores at a time step indicates the contribution of the left-out data item in the overall dataset at that time. It is also worth noticing that the naive approach to the leave-one-out strategy is computationally expensive. Therefore, to make this strategy scalable, we first apply the binning process before building the MST (as discussed before). Furthermore, we only leave-out the singleton bins (bins with only one data entry) and compute the contribution of these bins (i.e., their corresponding individual entries) to the overall outlying score. The reason is that if a bin contains more than one data entry, then the entries in that bin are not outlying.

To appreciate the importance of masking and swamping effects in abnormality detection, besides the concept of outlying data points, we provide a new concept called "inlying data points". That is, if we leave a data item out, then the outlying score of the resulted dataset increases or decreases, meaning this individual data point contributes to the overall outlying score. The decrement might mean that the data item itself is outlying, or its presence brings more outlying points (swamping effect). These points are called *outliers* in our approach. On the other hand, the score increment indicates that the data point hides the other outlying points if it exists in the dataset (masking effect). Such data items are called "inlying data points" or **inliers** in this paper. Figure 3 shows an example of **inlier** in the prevalence of human immunodeficiency virus (HIV) dataset collected from the University of Illinois at Chicago (UIC) repository [33]. The two variables are the prevalence of HIV female (ages 15-24) and male (in the same age range). In particular, the existence of *Lesotho* has masked the fact that *Botswana* is also significantly high on the prevalence of HIV. By removing *Lesotho*, *Botswana* now becomes an outlier as shown in the right panel of Figure 3. Hence, *Lesotho* is considered as the **inlier** of this scatterplot.

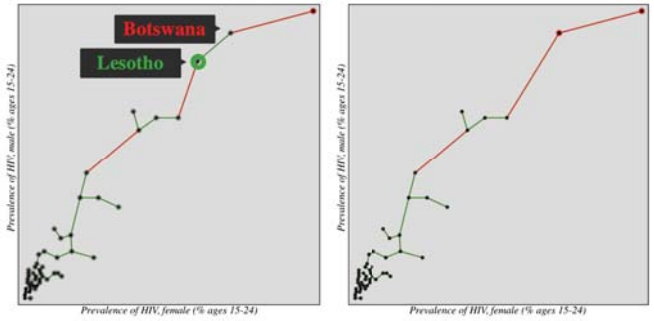

Fig. 3. An example of **inlier** in the prevalence of HIV data. By removing *Lesotho* in the left plot, *Botswana* becomes an outlier as shown in the right panel. *Botswana* is identified as **inlier** in our prototype.

Thus far, we show the MST example (Figure 2) and outlier/inlier example (Figure 3) in a 2D since it is easier to demonstrate the ideas in a two-dimensional space. However, the concept of outlier/inlier detection and our prototype can be naturally extended to multivariate outlier detection and visualization. The current MST distance metric is Euclidean. However, to avoid the "curse of dimensionality", other metrics (e.g., Manhattan distance *L1*) could also be explored [34] in the future research.

B. Visualization components and interactions

Figure 4 shows the main visualization components of our solution applied to a multivariate time series dataset collected from a high-performance computing center at a university [35]. The top panel contains a set of small multiples listed from left to right according to their timestamps. Each small-multiple is a radar chart plotting health metrics (such as CPU temperature, CPU load, memory usage, fan speed, and power usage) of 467 computing nodes in this center. The large numbers of curves (representing 467 nodes per radar chart) requires high rendering costs and produce visual cluttering issues. To overcome these issues, we only plot individual inlying and outlying entries as red and green curves correspondingly. Other normal computing nodes at each time step are aggregated into clusters using k-means algorithm [36] and visualize them as gray bands. This strategy also helps to highlight outlying/inlying data entries in the plot and thus is helpful in the abnormality detection process.

Figure 5 is an example of MTSAD radar chart representation to visualize nine essential variables at a specified time step in the dataset from the high performance computing center. The red curves are outlying computation nodes (one with low

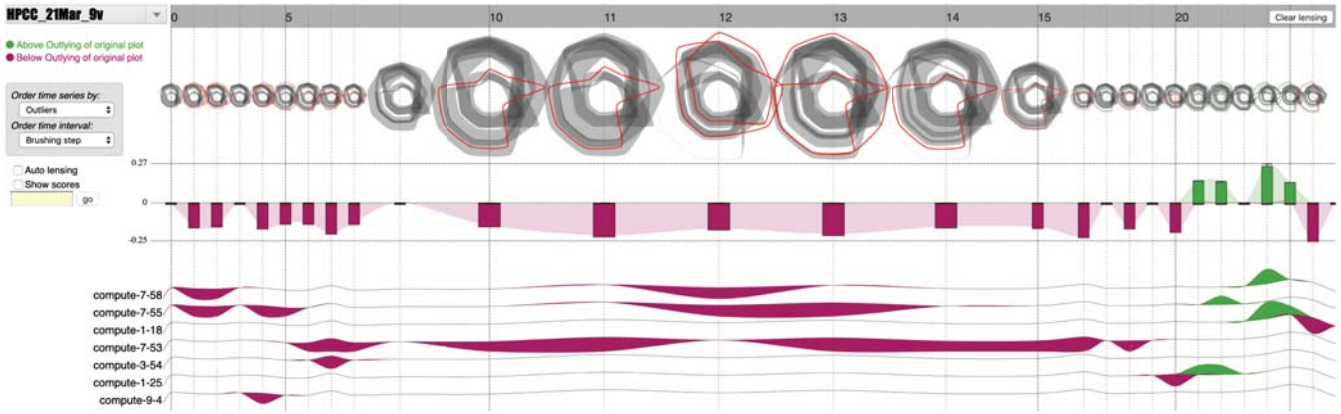

Fig. 4. Main visualization components of MTSAD applied in a multivariate time series dataset collected from a high performance computing center. These components include a time-line, a set of radar-chart small-multiples, the overview outlying box-plots, the item profile sections, and interactive options to assist the abnormality exploration process.

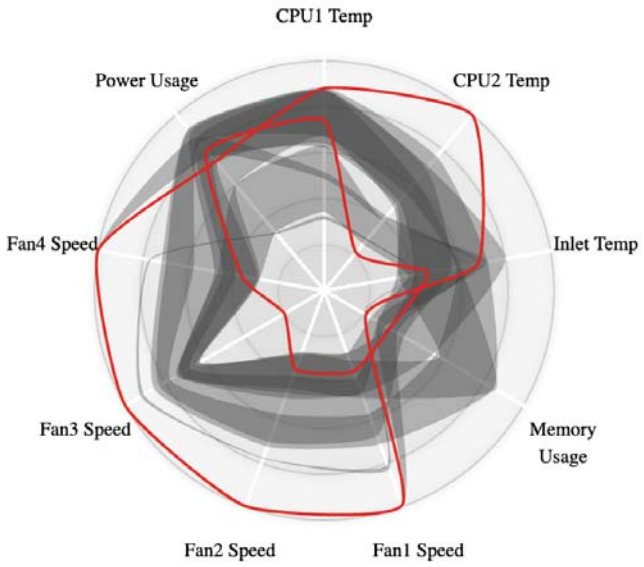

Fig. 5. MTSAD radar chart to visualize nine essential variables for 467 computation nodes from a high performance computing center. The red curves represent computation nodes with outlying data. The grey blobs represent clusters of normal nodes.

values and another one with high values for the monitoring variables). The grey blobs are the clusters of computation nodes with normal data values. It is observable that the red curves highlight the outlying data entries and indicates where the "hot spots" are at this time step.

Also, due to limited horizontal space, each radar chart is displayed as a small thumbnail by default. The default thumbnail size is just adequate for the analyst to have a good overview of the time series. On the other hand, MTSAD provides a timeline, which is also a lensing bar on top of the radar charts. Analysts can mouse over the time point of interest to expand the radar chart into a broader view for better details in case needed, as shown in the middle section of Figure 4. Also, the user can click on the thumbnail to bring out a detail view on demand (see fig. 7H).

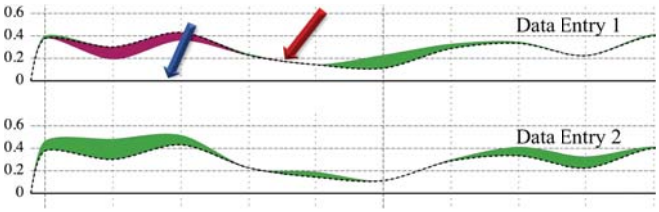

Fig. 6. The detail view in our MTSAD visualization of two example data entries. For instance, the solid line, at the blue arrow, is the base-line for the data entry (its position). The dashed line at the red arrow shows the outlying scores across the time of the multidimensional plots (the radar charts). The purple stream shows negative contributions, and the green one represents the positive contributions of the data entry to the overall outlying score of the multidimensional plots.

Figure 6 explains the view of outlying/inlying entries at the bottom panel of figure 4, called data entry (or data item interchangeably) profile section. Each data entry (e.g., *data entry 1* or *data entry 2*) is displayed at a baseline (e.g., at the blue arrow in the figure for *data entry 1*). Furthermore, for each data entry, there are purple/green stream graphs to represent outlying/inlying contributions of the data entry across time correspondingly (purple for negative contribution and green for positive contribution). These stream graphs stem from an outlying baseline (e.g., at the red arrow in the figure for *data entry 1*). This outlying baseline represents the overall outlying score (ranging from 0 to 1.0 exclusively) of the multivariate plot at a time step as a whole (i.e., without leaving any data entry out). The green/purple stream graphs show how much the outlying score increases/decreases compared to the overall outlying baseline if the data entry is left out. The data entry position and the outlying base-lines (at the blue and red arrows), and the outlying scores labels (on the left) are visualized in Figure 6 to help explain the visualization concept. However, they are removed in our solution to avoid visual cluttering issues.

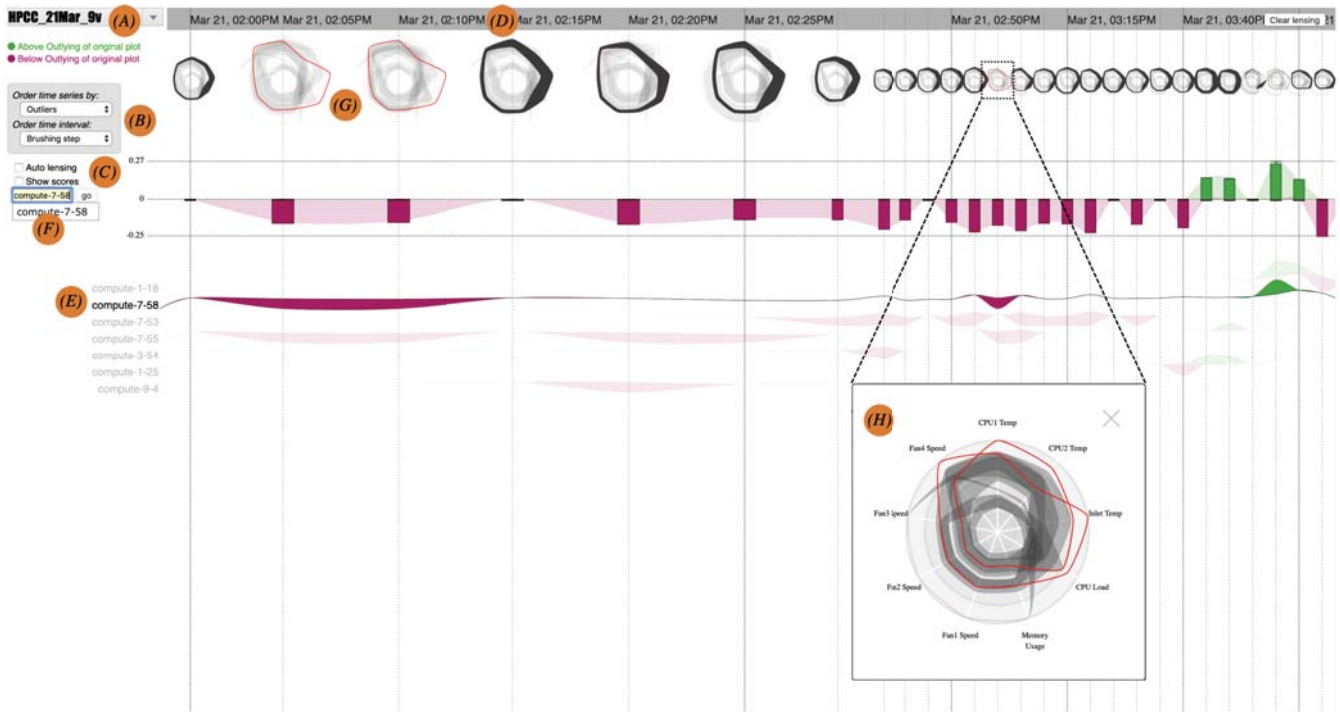

Fig. 7. MTSAD visual interface with the annotated interactive options from A to H. (A) Monitoring profile; (B) Ordering options; (C) Auto lensing and toggling scores options; (D) Lensing over time-line option; (E) Brushing on data entry option; (F) Search for data entry option; (G) Data evolution with small-multiples of radar charts; (H) A closeup view appears when user clicks a radar chart in the summary panel. Red curves are outiers, and grey blobs are clusters of regular data entries at each time step.

Also, as the aim of this application is to highlight the temporal inlying/outlying signatures of data entries. We filter out the other data entries which do not have significant contributions to the overall outlying scores of the datasets over time. This filtering operation also helps to reduce the visual load for users and therefore allows them to focus on data entries, which are more fluctuated in the time series. Furthermore, the data entries are ordered to bring the data entries with higher abnormality scores to higher places on the profile section to suggest investigation priorities.

To support the abnormality exploration process, MTSAD provides analysts with a rich set of interactive options [37], [38]. These interactions can be broadly classified into two categories as overall view options and on-demand details options (e.g., lensing, brushing, linking, and searching). The first category controls the overview of the system, and the latter allows the user to investigate details about a particular data item at a specific time step.

Belonging to the first category, MTSAD provides users an option to select a dataset to analyze from the dropdown at the top-left corner (see fig. 7A). There are also options to order the data entries in the data item profile section (see fig. 7B). These options are ordering by outlying/inlying scores ("Order time series by" options) over the brushing step only or all the lensing steps ("Order time interval" options). The scoring order options are by outlying score, or inlying score, or the sum of the absolute values of both (i.e., it brings the data items with the higher corresponding scores on top). The "Order time interval" options allow the user to control the order by using the corresponding score(s) over the brushing step only or overall the lensing steps. These ordering options enable the analyst to control how MTSAD organize the data entries as such to bring the items of interest to the top of the data entry profile section.

The application also provides analysts with options to do the lensing task automatically and show/hide outlying/inlying scores (see fig. 7C). The "Auto lensing" option, if selected, the application lenses automatically at the time step with the highest inlying/outlying or both (depending on the chosen option) score over the whole time series. It also brings the item to the top of the item profile panel. This functionality helps to point the analyst straight to where the "hot spots" are in the current multivariate time series to further investigate the potential abnormalities. The second checkbox ("Show scores") in this group allows the analyst to toggle the actual outlying/inlying scores for each data item over the whole time series. This option enables the analyst to quantitatively examine the actual outlying/outlying scores that MTSAD computes in its underlying computation process.

The second category of interactive options enables the on-demand details while exploring abnormalities using MTSAD. The first interaction type in this category is lensing. Whenever the analyst mouseovers a specific time step on the top timeline (see fig. 7D), the underlying radar chart and data entries at that

time step are scaled to provide better views about the data at the specified period. The scaling effect is also applied to the nearby items to assist the exploration task. The application orders the data entry profile section at the lensed period with the corresponding ordering options described above (see fig. 7B).

Besides, to investigate a specific data item, the analyst has options to mouseover that data entry in the item profile section (e.g., see fig. 7E) or to search for a specific entry using the search box (e.g., see fig. 7F). The system then highlights the data entry in the item profile section and fades out the others to let the user focus on analyzing the outlying/inlying score evolution across time for the specified entry. Similarly, small-multiples of radar charts also highlight the data item (as curves) or its corresponding clusters (as blobs) across time to allow the analyst to see how the data profiles evolve. For instance, it was outlying or belonging to some specific clusters (see fig. 7G). Furthermore, the analyst also has an option to click on any radar chart thumbnail to bring up the detail view of the chart for further investigation of the data at the specified time step. For instance, Figure 7H shows a detail view when the user clicks on the radar chart thumbnail at time step 12 of this dataset.

IV. Implementation

The solution is implemented as a web interface using JavaScript and D3.js [39]. Also, the leave-one-out computation process is expensive. Therefore, MTSAD splits this task into independent parts and executes them in parallel to save computation time. The current implementation uses web workers [40] to implement parallelisms. The source codes, demonstration pages, and videos are hosted on the Github page of the project: https://idatavisualizationlab.github.io/V/MultiOutliers/demos.html.

V. Use cases

To validate the use of MTSAD, we applied it in monitoring nine different essential variables for 467 computation nodes from a high performance computing center at a university. The following use-cases describe different situations where our monitoring solution is useful.

A. Use-case 1

The monitoring period for this use-case is on March 21, 2019, from 2:00 PM until 4:25 PM (five minutes every step and total of 30 steps). At 2:00 PM, the temperatures of the computation nodes began to rise. Fig. 8A is a details view of the nine monitoring variables at 2:20 PM. It is observable that several computation nodes were heating, and two nodes were outlying due to having high temperatures. Also, a few nodes started to sense the heat and increased their fan speeds. After that, the temperatures and fan speeds increased steadily. By 3:50 PM (see fig. 8B), it is visible that temperatures and fan speeds for many computation nodes were high. After that, at 4:25 PM, the system stopped receiving data from the high performance computing center.

Our system detected this suspicious behavior, and we reported this to the system administrators from the high performance computing center. They confirmed the issue and explained that it was due to the problem with the chilled water system (a system that is in charge of cooling the computation nodes at this center). Also, by 4:25 PM, they needed to make an emergency shutdown for all the computation and login nodes to avoid any further harm to the computing center before fixing the chilled water issue.

Furthermore, this use-case also proves the idea that inliers are equally important compared to outliers in the process of abnormality detection. For instance, it is obvious from Figure 8B that if we used the Box-Plot rule to compute outlying scores, there would be no outliers at 3:20 PM time step. Even though *compute-1-25* had low values and *compute-7-55* had high values for the monitoring variables compared to the others. Due to the masking effect, these two were not detected as outliers. However, leaving either one out in the outlying score calculation leads to higher outlying scores. Thus, the inlying score helps in this case to overcome the masking effect. In other words, the inliers enable the identification of potential outliers.

B. Use-case 2

The monitoring period for this use case was every one hour starting from June 23, 2019, at 01:00 AM until June 26, 2019, at 2:00 PM. Due to a long monitoring time and limited display space, in this case, the time-line switches to the time-step display (i.e., 0 up to 85) instead of the actual date/time display on the time-line. Figure 9 shows a sample output for this use-case when we mouse-over *compute-6-2*. MTSAD shows the evolution of the monitored variables for this computation node across time. It is observable that this computation node was constantly outlying due to the high values of temperatures and fan speeds. These constantly high values are harmful to the computation node, but the system administrator from this high performance computing center did not recognize this situation. MTSAD detected this harmful situation and brought *compute-6-2* to the top of the item profile section to suggest investigations. Also, when mouse over the *compute-6-2*, it was all represented as red, outlying curves from time step 0 up to time step 70 (i.e., from the start until 11:00 PM on June 25, 2019). We then reported this to the system administrators, and they discovered that this node was heavily scheduled during this period. They then rescheduled the loads for this computation node. Thus, from June 26, 2019, at midnight onward,*compute-6-2* became normal, and this computation node was clustered with other normal nodes (displayed as grey blobs for a group of nodes instead of a red curve as for outlying node).

Besides *compute-6-2*, MTSAD also brought *compute-1-26* to the top of the item profile section. This ordering event suggested us to investigate this case further. Fig 10 shows a view of MTSAD when clicking on the thumbnail of the radar chart at time step 4. This detail view shows that *compute-6-2* had a high outlying score due to high temperatures and

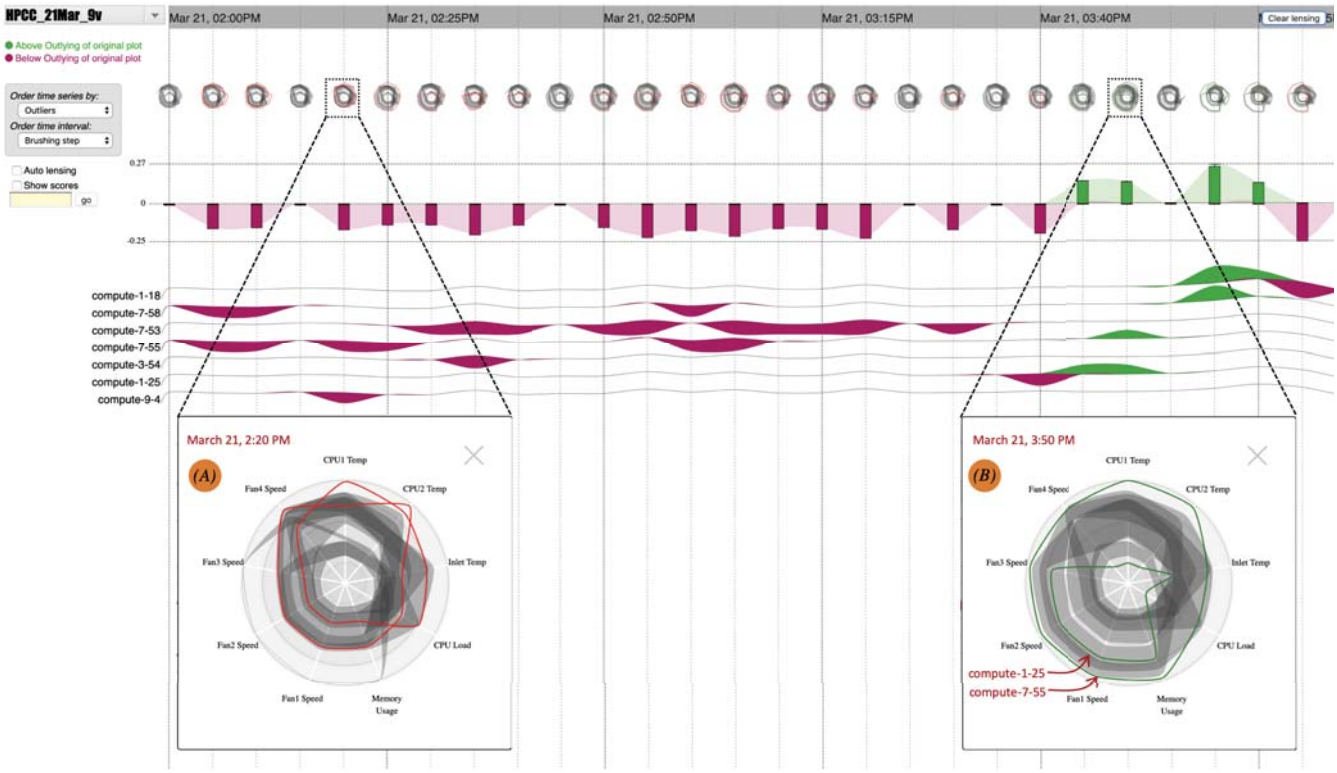

Fig. 8. Application of MTSAD in monitoring nine CPU-health related variables at a high performance computing center on March 21, 2019, from 2:00 PM until 4:25 PM and the chill water event. Panel (A), on March 21 at 2:20 PM, the temperatures of the computation nodes started to increase, and a few nodes sensed the heats and pumped their fan speeds. Panel (B), at 3:50 PM, many computation nodes were heated, and they increased their fan speeds.

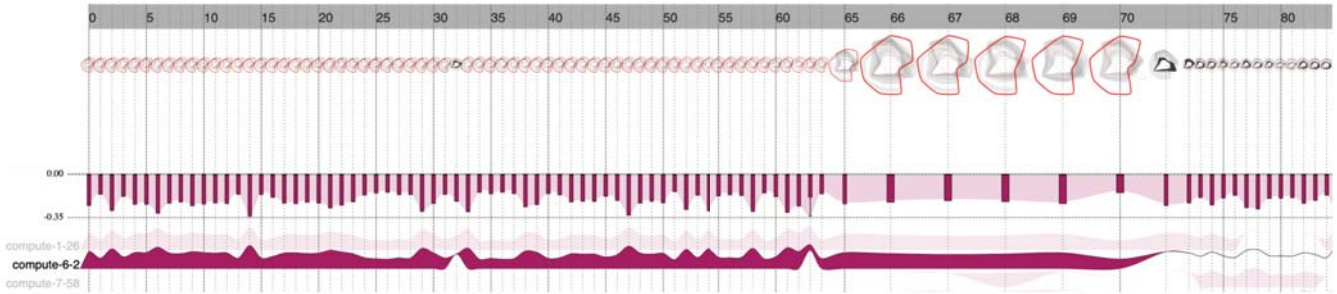

Fig. 9. Application of MTSAD in monitoring nine CPU-health related variables of 467 computation nodes at a high performance computing center from June 23, 2019, at 01:00 AM until 2:00 PM June 26, 2019. Mouseover *compute-6-2* highlights the evolution of its nine variables. In the summary panel, the red curve means it was outlying, and the grey blob means it was normal and belonged to a cluster (with other normal nodes) at a specific time step. Its stream-graph at the item profile section shows how much its outlying scores were over time.

fan speeds. On the other hand, *compute-1-26* was the top computation node in the item profile section due to its low values for the monitoring variables. This type of low-value outlier is equally important in this case because it shows that the computation node was underutilized while the other was heavily loaded. These two different types of outliers indicate that there is a need for finding better scheduling algorithms to balance the workload in this high-performance computing center.

Our solution received positive feedback from the center system administrators. They commented that MTSAD effectively detects and highlights the outlying/inlying computation nodes with high values for the monitoring variables. These high values suggest the system administrators take preemptive actions before the involved computation nodes become harmful. Equally important, it also provides warning signals via high outlying/inlying scores for computation nodes with low values for those variables. These high and low values indicate the inefficiency of the currently used load balancing algorithm. Please refer to the demo page at the Github page of our project to explore such use cases.

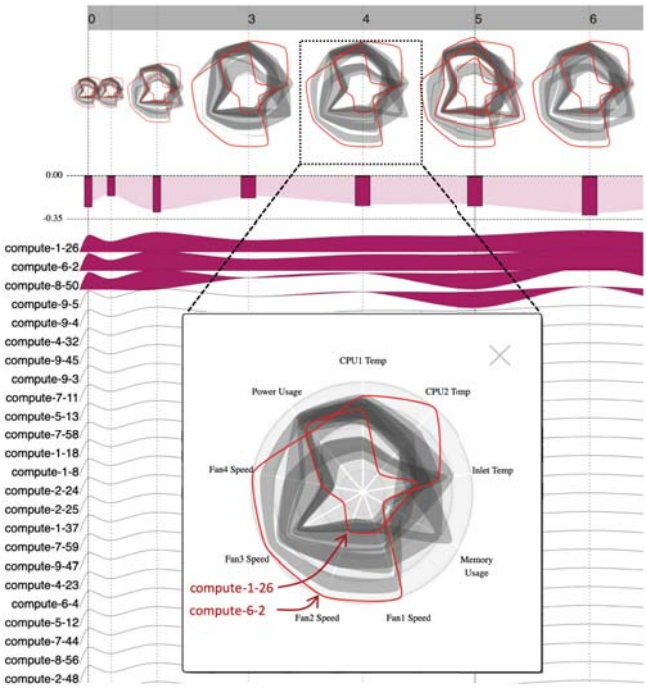

Fig. 10. Computation node *compute-6-2* is a node with outlying score due to its high values for some of the monitoring variables. On the other hand, *compute-1-26* has high outlying scores because of its low values for the variables. These different outlying types indicate load balancing issues.

VI. Conclusion

This paper presents a technique and an interactive interface for visualizing and analyzing high-dimensional time series data, which are becoming more and more popular. The technique adopts the leave-one-out validation metaphor into measuring the outlying score differences as the data item is temporarily knocked out from the current plot. The visualization presents data item temporal signatures by keeping track of the leave-one-out outlying changes overtime. The visual interface supports a wide range of interactive operations for exploring complex and large multivariate data, such as filtering, ranking, and lensing. This work is demonstrated on multivariate data generated from a high-performance computing center at a university. Predicting multivariate outliers based on the historical signatures of data entries is an interesting future extension of this work.

VII. Acknowledgments

The authors acknowledge the High-Performance Computing Center (HPCC) at Texas Tech University [27] in Lubbock for providing HPC resources and data that have contributed to the research results reported within this paper. The authors are thankful to the anonymous reviewers for their valuable feedback and suggestions that improved this paper significantly. This research is supported in part by the National Science Foundation under grant CNS-1362134, OAC-1835892, and through the IUCRC-CAC (Cloud and Autonomic Computing) Dell Inc. membership contribution.

References

[1] T. N. Dang, A. Anand, and L. Wilkinson, "Timeseer: Scagnostics for high-dimensional time series," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 3, pp. 470–483, March 2013.

[2] T. N. Dang and L. Wilkinson, "Timeseer: Detecting interesting distributions in multiple time series data," in *Proceedings of the 5th International Symposium on Visual Information Communication and Interaction*, ser. VINCI '12. New York, NY, USA: ACM, 2012, pp. 43–51. [Online]. Available: http://doi.acm.org/10.1145/2397696.2397703

[3] L. Wilkinson, "Visualizing big data outliers through distributed aggregation," *IEEE transactions on visualization and computer graphics*, 2017.

[4] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing*, vol. 262, pp. 134–147, 2017.

[5] J.-T. Chiang *et al.*, "The masking and swamping effects using the planted mean-shift outliers models," *Int. J. Contemp. Math. Sciences*, vol. 2, no. 7, pp. 297–307, 2007.

[6] A. S. Hadi and J. S. Simonoff, "Procedures for the identification of multiple outliers in linear models," *Journal of the American Statistical Association*, vol. 88, no. 424, pp. 1264–1272, 1993.

[7] V. Pham and T. Dang, "Outliagnostics: Visualizing temporal discrepancy in outlying signatures of data entries," 2019.

[8] V. Pham and T. Dang, "Cvexplorer: Multidimensional visualization for common vulnerabilities and exposures," in *2018 IEEE International Conference on Big Data (Big Data)*, Dec 2018, pp. 1296–1301.

[9] X. C. Chen, K. Steinhaeuser, S. Boriah, S. Chatterjee, and V. Kumar, "Contextual time series change detection," in *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM, 2013, pp. 503–511.

[10] V. V. Pham and T. Dang, "Mtdes: Multi-dimensional temporal data exploration system; strong support for exploratory analysis award in vast 2018, mini-challenge 2," in *2018 IEEE Conference on Visual Analytics Science and Technology (VAST)*, Oct 2018, pp. 100–101.

[11] J. Wilkes, "More Google cluster data," Google research blog, Nov. 2011, posted at http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html. Accessed: 2019-09-27.

[12] Alibaba, "Alibaba cluster trace program," Github, 2018, posted at https://github.com/alibaba/clusterdata. Accessed: 2019-09-27.

[13] H. Beyer, "Tukey, john w.: Exploratory data analysis. addison-wesley publishing company reading, mass. - menlo park, cal., london, amsterdam, don mills, ontario, sydney 1977, xvi, 688 s." *Biometrical Journal*, vol. 23, no. 4, pp. 413–414, 1981. [Online]. Available: http://dx.doi.org/10.1002/bimj.4710230408

[14] A. Luceo and D. Pea, *Autoregressive Integrated Moving Average (ARIMA) Modeling*. American Cancer Society, 2008. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470061572.eqr276

[15] L. Galante, "A Comparative Evaluation of Anomaly Detection Techniques on Multivariate Time Series Data," 2019.

[16] J. I. Takeuchi and K. Yamanishi, "A unifying framework for detecting outliers and change points from time series," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 4, pp. 482–492, 2006.

[17] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*. ACM, 1992, pp. 144–152.

[18] K. Heller, K. Svore, A. D. Keromytis, and S. Stolfo, "One class support vector machines for detecting anomalous windows registry accesses," *Workshop on Data Mining for Computer Security (DMSEC), Melbourne, FL, November 19, 2003*, 2003.

[19] F. T. Liu, K. M. Ting, and Z. H. Zhou, "Isolation-based anomaly detection," *ACM Transactions on Knowledge Discovery from Data*, 2012.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[21] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long Short Term Memory networks for anomaly detection in time series," *23rd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2015 - Proceedings*, no. April, pp. 89–94, 2015.

[22] G. Robertson, R. Fernandez, D. Fisher, B. Lee, and J. Stasko, "Effectiveness of animation in trend visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1325–1332, 2008.

[23] E. R. Tufte, N. H. Goeler, and R. Benson, *Envisioning information*. Graphics press Cheshire, CT, 1990, vol. 126.
[24] M. J. Saary, "Radar plots: a useful way for presenting multivariate health care data," *Journal of clinical epidemiology*, vol. 61, no. 4, pp. 311–317, 2008.
[25] A. Dasgupta and R. Kosara, "Pargnostics: Screen-space metrics for parallel coordinates," *IEEE Transactions on Visualization & Computer Graphics*, no. 6, pp. 1017–1026, 2010.
[26] N. V. T. Nguyen and T. Dang, "Ant-sne: Tracking community evolution via animated t-sne," in *Advances in Visual Computing*, G. Bebis, R. Boyle, B. Parvin, D. Koracin, D. Ushizima, S. Chai, S. Sueda, X. Lin, A. Lu, D. Thalmann, C. Wang, and P. Xu, Eds. Cham: Springer International Publishing, 2019, pp. 330–341.
[27] High Performance Computing Center. (2019) High performance computing center (hpcc) at texas tech university. [Online]. Available: http://www.depts.ttu.edu/hpcc/
[28] A. Inselberg and B. Dimsdale, "Parallel coordinates: A tool for visualizing multi-dimensional geometry," in *Proceedings of the 1st Conference on Visualization '90*, ser. VIS '90. Los Alamitos, CA, USA: IEEE Computer Society Press, 1990, pp. 361–378. [Online]. Available: http://dl.acm.org/citation.cfm?id=949531.949588
[29] U.S. Bureau of Labor Statistics. (2019) U.s. bureau of labor statistics. Accessed: 2019-09-27. [Online]. Available: https://www.bls.gov/home.htm
[30] M. Meyer, T. Munzner, and H. Pfister, "Mizbee: A multiscale synteny browser," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 897–904, Nov. 2009. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2009.167
[31] Kaggle, https://www.kaggle.com/datasets, September 2018, accessed: 2019-09-27.
[32] T. N. Dang and L. Wilkinson, "Transforming scagnostics to reveal hidden features," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 1624–1632, Dec 2014.
[33] A. Asuncion and D. Newman, "UCI machine learning repository," 2007, accessed: 2019-09-27. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html
[34] A. Anand, L. Wilkinson, and D. N. Tuan, "An l-infinity norm visual classifier," in *2009 Ninth IEEE International Conference on Data Mining*, Dec 2009, pp. 687–692.
[35] T. Dang, "Visualizing multidimensional health status of data centers," in *Programming and Performance Visualization Tools*, A. Bhatele, D. Boehme, J. A. Levine, A. D. Malony, and M. Schulz, Eds. Cham: Springer International Publishing, 2019, pp. 273–283.
[36] J. Hartigan, *Clustering Algorithms*. New York: John Wiley & Sons, 1975.
[37] R. Amar, J. Eagan, and J. Stasko, "Low-level components of analytic activity in information visualization," in *Proc. of the IEEE Symposium on Information Visualization*, 2005, pp. 15–24.
[38] S. F. Roth, J. Kolojejchick, J. Mattis, and J. Goldstein, "Interactive graphic design using automatic presentation knowledge," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 1994, pp. 112–117.
[39] M. Bostock, V. Ogievetsky, and J. Heer, "D3 data-driven documents," *IEEE Trans. Vis. Comput. Graph.*, vol. 17, no. 12, pp. 2301–2309, 2011.
[40] Mozilla and individual contributors. (2018) Web workers api. Accessed: 2019-09-27. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API