Erschienen in: IEEE Transactions on Visualization and Computer Graphics ; 22 (2016), 1. - S. 141-150
https://dx.doi.org/10.1109/TVCG.2015.2467553

Temporal MDS Plots for Analysis of Multivariate Data

Dominik Jäckle, Fabian Fischer, Tobias Schreck, Daniel A. Keim

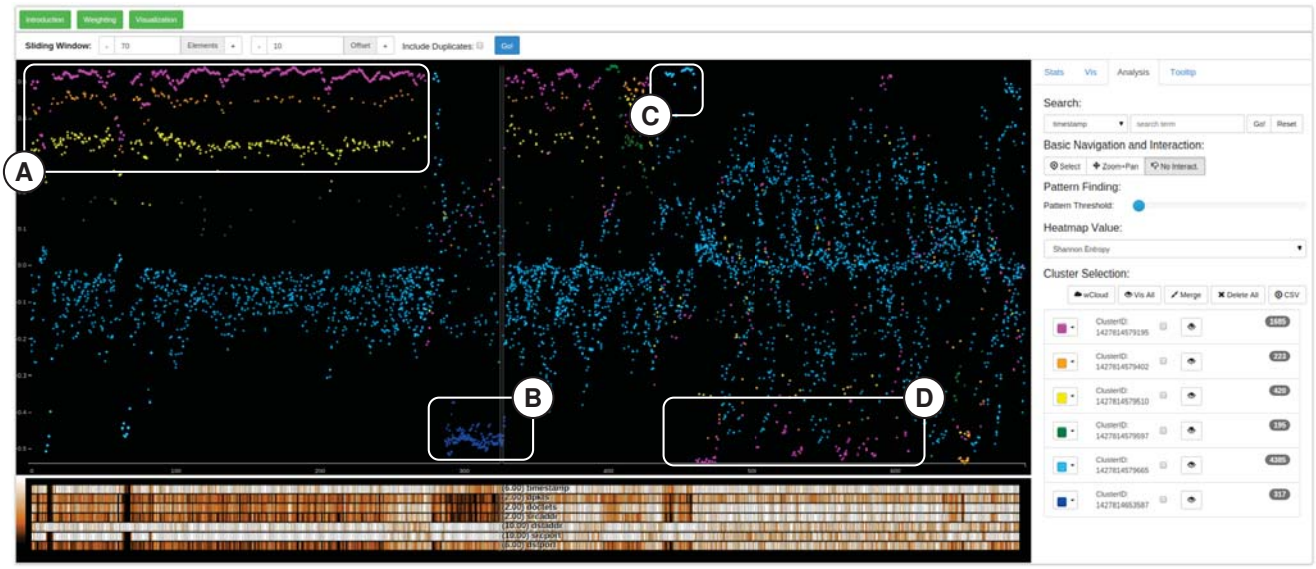

Fig. 1. Temporal MDS plots (top) applied to network traffic data, which was collected from a /16 computer network over a period of 24 hours. For each temporal MDS plot the sequentially aligned matrix (bottom) provides an overview of correlations among dimensions. The visualization reveals a distributed brute-force attack (A, D) and various different port scans (B, C).

Abstract— Multivariate time series data can be found in many application domains. Examples include data from computer networks, healthcare, social networks, or financial markets. Often, patterns in such data evolve over time among multiple dimensions and are hard to detect. Dimensionality reduction methods such as PCA and MDS allow analysis and visualization of multivariate data, but per se do not provide means to explore multivariate patterns over time. We propose Temporal Multidimensional Scaling (TMDS), a novel visualization technique that computes temporal one-dimensional MDS plots for multivariate data which evolve over time. Using a sliding window approach, MDS is computed for each data window separately, and the results are plotted sequentially along the time axis, taking care of plot alignment. Our TMDS plots enable visual identification of patterns based on multidimensional similarity of the data evolving over time. We demonstrate the usefulness of our approach in the field of network security and show in two case studies how users can iteratively explore the data to identify previously unknown, temporally evolving patterns.

Index Terms—Multivariate Data, Time Series, Data Reduction, Multidimensional Scaling

1 Introduction

Today's world is driven by the continuous collection of data from various domains: Computer networks, healthcare, and finance markets are prominent examples. Particularly in the exploration phase, a key task in understanding complex data is to group it into a set of discernible areas (patterns). Real-world data is often multivariate and evolves over time, posing a challenge to detecting such patterns visually. Visual analytics aims to support understanding of complex data and finding patterns. It suggests that analysts are involved into the automatic data analysis process by steering analysis parameters and exploration of data by visualization. As a result, analysts are supported to understand data and draw conclusions for a specific analysis task.

- *Dominik Jäckle, Fabian Fischer, and Daniel Keim are with the University of Konstanz, Germany. E-mail: {forename.lastname}@uni-konstanz.de.*
- *Tobias Schreck is with Graz University of Technology, Austria. E-mail: tobias.schreck@cgv.tugraz.at.*

For example in the field of network security, threats show different and often very dynamic behaviors. In so-called port scans, attackers explicitly search for open ports to receive access to a given system. However, such computer threats constantly evolve over time and change their behavior. Large amounts of data need to be analyzed without any prior knowledge about which threats to expect, and when. Existing approaches such as supervised machine learning, typically rely on classifiers and thus prior knowledge to detect or predict patterns. Visual data analysis is promising in helping explore and analyze data in an unsupervised way. Prominent examples of successful high-dimensional visualization techniques include e.g., parallel coordinates and glyph-based techniques. However, many high-dimensional data visualization techniques cannot directly consider the temporal aspect, or only for selected single dimensions. The so-called time series path techniques and others (see Section 2) consider multivariate temporal data, but lack presenting temporal multivariate data with multiple events at a time.

We propose Temporal Multidimensional Scaling (TMDS) which takes temporal multivariate data into account and visually presents the data enabling analysts to identify patterns and explore the data space, following the visual analytics process. Our approach is based on two driving questions: Firstly, how to process and visualize tem-

poral multivariate data to allow analysts explore patterns? Secondly, once a pattern has been identified, how can we automatically find similar patterns? TMDS applies a sliding window approach on the data and computes a one-dimensional (1D) MDS for each window. The resulting sequence of 1D MDS mappings are then organized along the temporal axis: The x-axis represents the time, and the y-axis represents the MDS similarity value. Similar events are grouped over time and can efficiently be identified. To analyze the multivariate nature, we augment the visualization with a sequenced diversity matrix aligned with the MDS plot revealing the different temporal behaviors of single variables. Furthermore, we introduce a new algorithm to find similar patterns based on the user selection and the behavior along dimensions. TMDS enables the efficient detection of recurring patterns and further allows to identify evolution of patterns, being based on varying scales and intervals. Another technical contribution of our approach is an appropriately aligned visualization of the sequence of 1D projections. Most existing approaches employ two-dimensional MDS projections. TMDS relies on 1D MDS projections, taking the second dimension in the plot to show the change of multivariate data patters over time.

The remainder of this paper is organized as follows. First, we discuss related work in Section 2. Then, we give a brief example in Section 3 about entailed benefits of temporal 1D MDS plots and how analysis is performed using our prototype. In Section 4, we provide a three-step pipeline to derive TMDS. In Section 5, we propose two extensions, which enable the user to visually and automatically identify patterns. We further show the usefulness of our approach in a case study with application to network security in Section 6 and provide a discussion of results in Section 7. Section 8 concludes.

2 Related Work

We briefly discuss related work from multivariate data mining, visualization, and visual analysis of temporal multivariate data.

2.1 Multivariate Data Mining and Visualization

Multivariate data analysis methods consider several dimensions (variables) simultaneously. Typically, dimensions in multivariate datasets are related and cannot be considered independently [32]. This is different with respect to multi-dimensional data, where individual dimensions are orthogonal to each other and may be reduced e.g., by feature selection techniques [31]. State of the art methods for the automated analysis of multivariate data can be found in the domain of machine learning [35, 18]. Supervised machine learning techniques require a priori knowledge about the data patterns to be searched, such as a classification structure to segment data. On the other hand, unsupervised techniques are suitable for the analysis of datasets whose patterns are not known in advance, e.g., including clustering analysis.

Many techniques support visualization for multivariate data, including methods like geometric projections (Parallel Coordinates [21], Andrews curves [3], Star Coordinates [24]), pixel-oriented techniques (Recursive Pattern [26], Pixel Bar Charts [27]), hierarchical displays (Dimensional Stacking [30]), or glyph-based techniques (Chernoff faces [8], Star glyphs [6]). Additional examples for the visualization of multivariate data can be found in the survey by Kehrer and Hauser [25]. Typically, these visualization approaches do not explicitly consider the temporal aspect of multivariate data. Either the temporal behavior is visualized only for single dimensions, or a statistical aggregate of the multivariate variables. A common approach to combine both, the temporal as well as the multivariate aspect, is the application of small multiples [39]: This way, multivariate visualizations can be tracked over time. However, visualizations are sequenced forcing the user to split perception attention which can impede accurate identification of temporally evolving patterns. An extensive survey of other visualization techniques for time-oriented data is found in [1].

2.2 Dimensionality Reduction Techniques

Analysis and visualization of high-dimensional data is a difficult problem. The so-called curse of dimensionality [19] impedes the ability to compactly visualize and identify patterns in multivariate data. Dimension reduction techniques target to detect and consider only interesting dimensions and their relation to each other for analysis. For example, Self-organizing Maps (SOM) [29] can be utilized to group data based on their similarity to each other into a predefined layout. Principal Component Analysis (PCA) [23] and Multidimensional Scaling (MDS) [9] among others, are not tied to producing 2D layouts, but project the data into a predefined n-dimensional space. Results represent a linear or non-linear combination of the original dimensions [32]. Dimension reduction techniques are widely used for the analysis and visualization of multivariate data. Yet, multivariate data is often temporally evolving and the dimension of time needs to be considered.

2.3 Time-Dependent Dimensionality Reduction

Dwyer et al. [12] take first steps and propose to map time to the third dimension of a two dimensional baseline MDS plot. Conceptually similar to a space-time-cube, changes over time can be tracked in a 2.5D view. Several approaches have been presented [33, 20, 40, 4], which project multivariate data to two dimensions. The idea is, that a single data entry is tracked over time, and the path of such single entry is visualized in the resulting projection. While these techniques may allow to detect e.g., cyclic patterns, the displays may also quickly lead to cluttered structures, as the data items may be plotted to arbitrary (x,y) coordinates and following them in 2D is a perceptionally difficult task. Crnovrsanin et al. [10] therefore discuss the usage of 1D plots for temporal analysis of movement data using PCA.

2.4 Delineation to our Work

The aforementioned methods track time for single data records (entries in a data table) that give rise to a temporal behavior in a 2D plot with continuous coordinates. Existing approaches typically consider a discrete projection for every time point, which may include abrupt changes in the (x,y) position of the data item location. In our approach, we propose a temporally smoothed version of time-dependent multivariate plots, by considering a sliding window approach with overlap to do the projection. Thereby, the resulting temporal patterns show higher smoothness, which acts as a filter to suppress abrupt changes which however, may only have very limited local support. Our approach is able to emphasize temporal evolving patterns of multivariate data, whose entries are not associated to each other but possibly share a common behavior across several dimensions. By using a 1D layout, we also introduce a simpler linear structure (as opposed to 2D trajectories), which analysts can easily follow and link to data details. As we show in the case studies (Section 6), this approach allows to identify patterns at various resolutions and interval sizes, which can be effectively tracked by the analyst.

3 Basic Idea of Temporal MDS

Our TMDS approach follows a visual analytics process and allows the user to browse the data and to refine parameters and inputs. TMDS is suitable for the analysis of any temporal evolving multivariate data, either numerical or categorical. Our approach first allows the analyst to select and weight single dimensions of a multivariate input data set, according to their importance. Then, TMDS applies a sliding window of given size and overlap to the time-dependent data. For each window, a 1D MDS analysis is performed. This way, similar entries are grouped accordingly over time, based on the correlation of dimensions. The analyst may also adjust the weighting of the dimensions, so that dimensions of interest can be prioritized and the task requirements are met. The re-computation of the MDS may reveal several temporal aligned patterns, as depicted in Figure 1. They are spatially separated along the y-axis (similarity) but still evolve over time (x-axis). The patterns are presented as groups of entries that evolve over time and are aligned on the same similarity level; they can be validated in combination with the diversity matrix plotted below the MDS. The diversity matrix is aligned with the window and quantitatively presents computed diversity indices among dimensions through color. It helps the analyst to efficiently understand correlations between dimensions

and thus to draw conclusions. The analyst further selects a salient pattern and runs the algorithm to find similar patterns. Similar patterns are then visually highlighted and separated through color, and listed next to the visualization. Using visual analytics, the analyst re-configures TMDS utilizing identified similar patterns as new input. This way, data can be explored based on new insight. An example result is presented in Figure 1. The selection is highlighted in magenta. The prototype provides details on demand and thus allows the analyst to inspect the raw data of the selection, which reveals a distributed brute-force attack to different servers. On demand, found pattern data can be exported as input for subsequent analysis tools.

4 Temporal Multidimensional Scaling

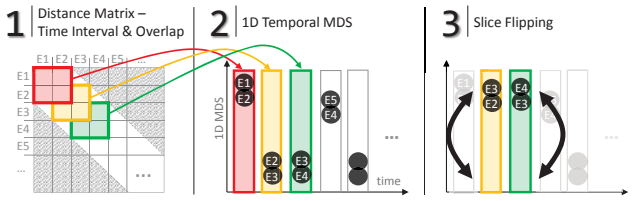

Fig. 2. Consecutive three-step pipeline for the TMDS computation. A dataset comprises the entries E_1 to E_N (each entry holding multiple dimensions), which are processed in temporal ascending order. (1) Based on the weighted distance matrix, a sliding window with overlap is applied and (2) 1D MDS computed for each window separately. The result is sequentially aligned on the time axis. (3) Because MDS is not invariant to rotation, we apply a slice flipping heuristic.

Our Temporal Multidimensional Scaling (TMDS) approach computes aligned temporal 1D MDS plots for multivariate data. We apply a sliding window to the temporal sequence of multivariate data and compute the MDS for each slice separately. Note that this approach is applicable to sequential as well as temporal data and therefore requires the data to be loaded in ascending temporal order, making it also suitable for real-time applications. The results are then plotted sequentially along the time axis.

MDS is a well-known dimension reduction technique, used to preserve similarities across multivariate data. Compared to e.g., PCA which requires co-variances, MDS requires a distance matrix as input, which defines the similarity between entries. Distances can be computed for various data types, including numerical distance between numerical values, binary distance between strings, cosine distance between documents in vector space, among others. Hence, MDS suites our needs of handling categorical data, which we encounter for instance in network security data (Section 6), healthcare data, and finance data. We compute the distances for categorical data as follows:

$$distance(A,B) = \frac{\sum_{i=1}^{|dim|} [A_i \neq B_i] \cdot w_i}{|dim|} \quad (1)$$

The distance or dissimilarity between two entries A and B is computed by first iterating all dimensions from $i = 1$ to the total amount of dimensions $|dim|$. Using Iverson Brackets [17], the i-th dimensions of the entries are compared with each other and the result is multiplied with dimension weight w_i. We then compute the average by dividing by the total amount of dimensions to derive the final distance value.

Weighting single dimensions has direct impact on the similarity calculation and thus the 1D MDS plot. We allow the user to define the weights w_i for each dimension individually. Especially domain experts typically know dimensions that are from interest for certain tasks at hand and can make use of their domain knowledge to influence the computation and the exploration of temporal patterns. The step of weighting becomes tedious if the data includes loads of dimensions. Hence, our approach supports predefined weightings, which are presented as suggestions. Suggestions are predefined but facilitate and speed up the analysis if combinations of dimensions reoccur among different datasets. Weighting dimensions is possible in the range of $[0,1]$. The lower the weight is, the less impact the weighted dimension has in the MDS analysis. For example, a dimension weighted with 0 is excluded from the computation of the distance matrix. To implement TMDS, we define the following three-step algorithm (Figure 2):

1. **Sliding Window:** Run sliding window with overlap and user-defined parameters along the sequence of data items, and compute the distance matrix for all entries in the given window.
2. **Temporal 1D MDS:** Apply multidimensional scaling to the distance matrix of each window step. The outcome of each computation is a one-dimensional ordering of the multivariate records and the basis for the sequential visualization.
3. **MDS Slice Flipping:** Multidimensional scaling is not invariant to rotation. Similarity values upon multiple TMDS computed slices may not share same similarity positions but be rotated by 180 degrees. As a result, temporally evolving patterns may not be clearly identifiable. A canonical orientation of the 1D MDS orderings needs to be obtained. We propose to use a heuristic based on the orientation of the entries contained in the overlap. Salient patterns are marked in Figure 3 (3), which were not equally clear without our flipping test.

Following, we will give an in-detail overview of these steps. Starting in Section 4.1, we discuss the sliding window, followed by 4.2, where we introduce TMDS and slice flipping.

4.1 Sliding Window

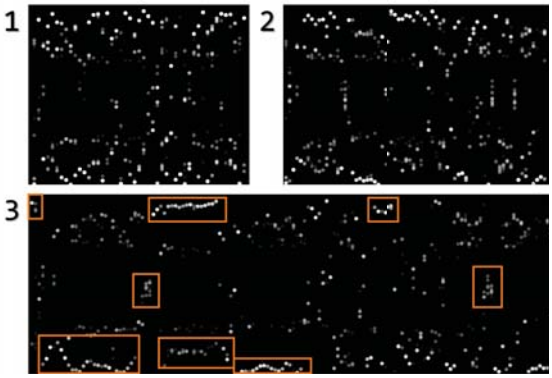

Fig. 3. Sliding window approach applied to a multivariate dataset with 15 dimensions and 1420 entries. For a window size of 30 entries, we applied an overlap of (1) 0 entries, (2) 10 entries, and (3) 20 entries. The bigger the defined overlap, the more slices are computed, resulting in a visualization whose layout becomes stable and reveals salient patterns.

To find temporally evolving patterns within multivariate data, we apply the sliding window approach to the data and compute the distance matrix for each window separately, serving as input to the 1D MDS computation. The sliding window approach is suitable, because it sequentially processes the data taking its temporal behavior into account. This way, patterns are connected step-wise (window-wise) and do not require additional cognitive load as it is the case using for example small multiples, among other spatially disconnected visualizations.

Suppose we compute the MDS for several windows without any overlap. Our result is one-dimensional, which means that the dimension with the highest variance has the highest impact on the result. In this scenario, the windows do not share any relation by means of reused entries. This means, for each window the dimension with the highest variance can be a different dimension, which results in an unstable layout. Hence, it would be harder to identify possible patterns. Figure 3 shows the effect of the sliding window approach applied to a dataset that contains 1420 entries with a window size of 30 entries.

(1) shows the result of TMDS without any overlap. As expected, the sequence of 1D projections is not smooth but fluctuates significantly. With increasing overlap, more windows are computed. (2) shows the same result, but with an overlap of 10 entries. Patterns do already stand out prominently. In (3), the overlap is 20 entries, even further increasing the stability, clearly showing evolving temporal patterns. Figure 5 (3) shows the path of reused entries that are included in the chosen overlap. Reused entries evolve on the same similarity level not causing a distorted perception of patterns. Adding overlap to the sliding

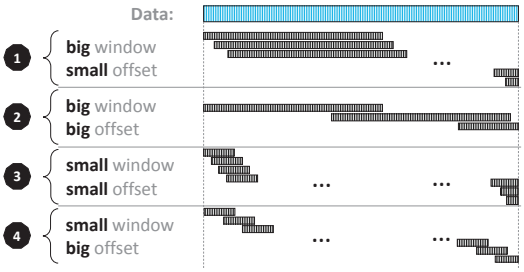

Fig. 4. Distinction of cases with respect to window size and offset.

window clearly results in smooth transition in the patterns. In the final visualization, reused entries are discarded and each presented slice contains the same amount of data entries according to the window size. The complexity of computing the distance matrix for n entries and m dimensions lies in $\mathcal{O}(n^2 \cdot m)$ in worst case, meaning that the chosen window covers all entries. The application of a classical MDS per window results in $\mathcal{O}(n^3)$ due to the expensive step of performing the eigendecomposition [41]. Thus, the overall complexity lies in $\mathcal{O}(n^3)$. However, in practice the window size is not n and thus the computation time differs with respect to window size and offset. Figure 4 depicts the distinction of cases with respect to window size and offset. Offset refers to the amount of entries a window is moved – small offset implies high overlap. Following, we outline the four derived cases with respect to Figure 4: (1) A big window size and a small offset results in high computation time, but provides a smooth transition in the patterns. (2) A big window size combined with a big offset requires high computation time and possibly does not provide smooth transitions. (3) Applying a small window size and a small offset still requires high computation time due to the amount of computed windows. However, transitions in the patterns are likely to be even smoother compared to case (1), because of the small window size and the reduced amount of considered entries per slice. (4) In contrast, the application of a small window size and a big offset reduces the computation time, but does not provide smooth transitions in the patterns.

We assume it is desired to have low computation time and smooth temporal slices. A big window size results in increased computation time, since both distance matrix and MDS are computed for an increased amount of entries. Then choosing a small offset increases the computation time even more. We argue that regarding the distinction of cases, cases (3) and (4) are most promising with respect to computation time and offset. For now, we use pragmatic rule-of-thumbs to set the offset, e.g., using 10% of the window size with good results. Automatically finding appropriate overlap sizes is left to future work, with one idea to use a visual salience function to search automatically for parameters that show interesting visual results. Techniques based on Hough transform analysis or other interest measures may be a starting point to this end [38]. A drawback of smoothing is the possibly hiding of outliers. By increasing the window size and thus adding data entries to a slice, former outliers are likely to join a cluster if the discrepancy decreases among entries. However, outliers that show high discrepancy to all other entries are preserved.

4.2 Temporal 1D MDS and Slice Flipping

Based on the sliding window, the MDS is computed for each window separately and then sequentially aligned in the Cartesian coordinate

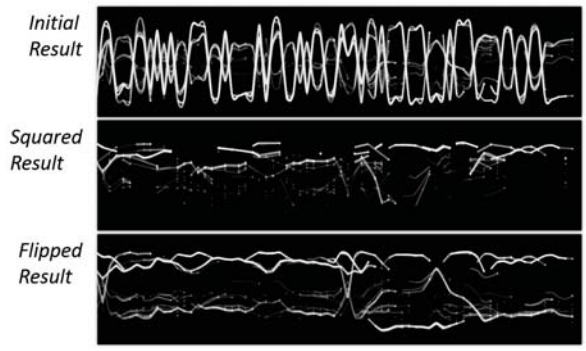

Fig. 5. The visualization shows the paths of reused entries of subsequent MDS slices. (1) shows the initial temporal result. MDS is not invariant to rotation. We therefore propose two heuristics to the result: (2) Square all results to level them. (3) Flip slices if more than half of reused entries change their leading sign in the subsequent slice.

system. The x-axis is the time and the y-axis is the 1D similarity value derived from the MDS computation. The numerical range of similarity values along the y-axis is always restricted by the bounds $[-0.5, +0.5]$. This is due to the fact that the computation of distances (see Equation 1), including weighting, is normalized to the range of $[0, 1]$. We created a pathline visualization to test how good temporal patterns evolve over time; it visualizes the paths of data entries (overlap) separately. Figure 5 (1) shows the initial result. As illustrated, the pathline alternates between positive and negative values within the bounds $[-0.5, +0.5]$. An alternating leading sign for a majority of reused entries (indicated by alpha-blending) in subsequent slices indicates that also co-located points are likely to alternate their leading sign, meaning that possible temporal patterns are interrupted. Patterns evolve on opposite sites instead next to each other. A reasonable solution is presented in Figure 5 (2): all results are squared leading to a leveling of the results and thus patterns. Squaring the results has the side effect that adjacent mingled patterns are additionally spatially separated. However, squaring the same negative as well as positive value leads to an overplot of patterns, because they are leveled to the same positive position. This is why, we do not want to discard negative values. Therefore, we decided to check subsequent slices, if the leading sign of reused entries changes. If more than half of the reused entries switch their leading sign, we change the leading sign of all entries contained in the subsequent slice (slice flipping). Thus, we compare the next but one slice to the current subsequent slice, and so on. Using a lower or higher threshold for flipping might not improve the visualization in quality, because either too few or many slices are likely to flip. The result of our heuristic is presented in Figure 5 (3), with the effect that additional temporal patterns become visible.

4.3 Details and Scalability

Our prototype runs as web application in a client server environment. The computation of the distance matrix as well as of the MDS require high runtime and thus are computed on the server-side. Our approach is based on a sliding window, applied to the distance matrix, which enables us firstly, to reduce the computation of similarity values regarding the window size, and secondly, to parallelize the MDS computation. As a result, we have a fully parallelizable approach allowing an interactive exploration of the data. Given the input of a temporal ascending sorted list of multivariate entries, we apply the sliding window directly on the dataset. Due to the symmetry of the distance matrix, one thread can compute only one diagonal of the distance matrix for the given window. Then, the one-dimensional MDS is computed for the distance matrix using the MDSJ Java package [2]. We note that in our implementation, in computing the MDS for a given window step, we reuse already computed distances from previous steps according to the overlap, and only compute anew distances, which stem from the newly considered data entries. It might also be possible to con-

sider incremental projection schemes which provide further speedup, but leave this for future work. The final flipping of the slices takes place in linear time and therefore does not require optimization. However, we use SVG on the client-side, which restricts the amount of data being interactively visualized. In order to also increase the visual scalability, the client-side can be replaced with efficient representations such as WebGL, among others.

5 Visual Similarity Search and Pattern Finding

The TMDS visualization provides an aggregate view to a time series of multivariate data. Specifically, the sequence of 1D MDS plots shows gradual changes of 1D distributions of entries, reflecting on the similarity relationships of the data entries and their change over time. However, the 1D plots do not show the behavior of the underlying dimensions. This detail information is needed in two cases: Firstly, if patterns are visually separated, the user needs details on the dimensions to find possible explanations of the MDS pattern. Secondly, increasing the amount of dimensions can result in less prominent visual patterns; having information on correlations of dimensions helps to interactively select (reduce) individual dimensions and facilitate the analysis by excluding potentially irrelevant dimensions. Therefore, we provide facilities to manually brush a data region and browse corresponding entries to find correlations among dimensions. However, this task is challenging even for a small set of dimensions, because various dimension permutations need to be taken into account. Following, we introduce two techniques: A visual approach using a matrix that visualizes correlations among dimensions and windows, and an automatic approach to find similar patterns, based on a user selection.

5.1 Visually Identifying Patterns

The visual identification of multivariate patterns is mapped to identifying salient patterns within the TMDS visualization. To support the process of finding patterns also dimension-wise, we introduce a diversity matrix as a heatmap, which is displayed below the TMDS plot, aligned with the sliding window. In this heatmap, each column corresponds to one window and each row to one dimension.

Diversity quantitatively reflects the amount of differing types or values within the dimensions and can be computed in various ways. It helps the user to draw conclusions based on the diversity correlations which describe the dimensions. We implemented two information theoretic measures to assess the diversity of dimension values per window. Considering that, we determine the different categories (following referred to as i) of values per dimension by binning. The **Shannon Entropy** H is computed as follows [36]:

$$H = -\sum_{i}^{n} p_i \cdot log_2(p_i) \tag{2}$$

According to the definition of the Shannon Entropy, p_i describes the proportion of a character i occurring in a string. Applied to our scenario of having multivariate categorical data, p_i describes the probability of category i occurring within a dimension. In contrast, the **Simpson Index** D is computed as follows [37]:

$$D = 1 - \sum_{i}^{n} \frac{m_i \cdot (m_i - 1)}{m \cdot (m - 1)} \tag{3}$$

m_i describes the amount of occurrences of category i and m the total amount of the categories per dimensions. We apply *min-max* normalization to color code single diversity values with respect to all dimensions and windows. The used colormap maps low diversity to black and high diversity to white along a brown gradient. Figure 6 shows the use of both the Shannon Entropy and the Simpson Index diversity measures for a test dataset of the domain of network security, containing approximately 15.000 entries and 16 dimensions. We observe high correlation between the Entropy and the Simpson index in our applications. While we implemented these two in our prototype, additional information theoretic measures can be chosen, depending on task and data. The diversity heatmaps provide an overview of the diversity of attributes and their changing over time. It is useful for identifying correlated dimensions for dimension filtering. At the same time, it is useful to compare changes in the MDS plot with changes in the diversity across the data dimensions, for exploring a) interesting time slides, and b) obtaining starting points for explaining the MDS patterns by properties of the underlying multivariate data.

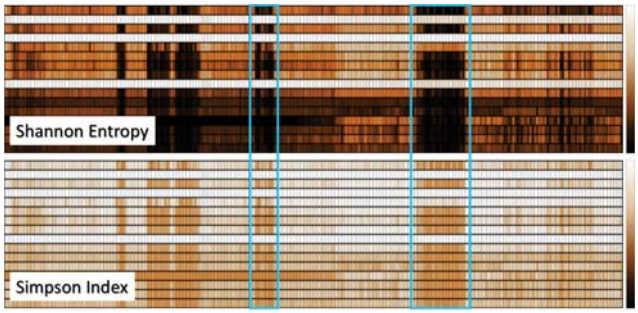

Fig. 6. Diversity Matrix. Application of Shannon Entropy (top) and Simpson Index (bottom). Columns are temporally aligned with the TMDS. Diversity is mapped to color (black is low diversity and white indicates high diversity) and reveals correlations between dimensions (rows).

5.2 Finding Similar Cohorts

In the previous Section 5.1, we described how patterns can be visually identified using our proposed TMDS in combination with the diversity matrix. Salient patterns that can be identified using the TMDS, typically consist of several data entries building a temporal cohort. This means, the entries share alike similarity values over a certain time period. Such patterns can be salient during one time period, but hidden during another time period. Patterns are declared as hidden if they correlate with other patterns and/or are distributed within other patterns. This effect can occur because of the sheer amount of other data entries which can influence the projection. To reveal other related patterns, we offer the user the option to manually select corresponding data entries, based on which related cluster of entries are automatically computed and highlighted. We argue that using an approach similar to the computation of the distance matrix (proposed in Section 4), will reveal similar patterns aligned with the layout of the TMDS.

Function *findSimilarPatterns*$(\mathbb{D}, \mathbb{S}, threshold)$
  $d \leftarrow 0$
  $similarities \leftarrow []$
  foreach *entry* E *in* $\mathbb{D}$ **do**
    $d \leftarrow distance(E, \mathbb{S})$
    $similarities \leftarrow sortedInsert(d, similarites)$
  end
  return $1D\_DBSCAN(similarities, threshold)$
End

Algorithm 1: Find similar patterns.

Algorithm 1 provides a short overview on how similar patterns are found. Input are the overall data $\mathbb{D}$, the selection $\mathbb{S}$ and a user-defined *threshold*. We follow two consecutive steps: Firstly, calculate a distance value per entry and user-defined selection. Secondly, sort all distance values and cluster them. We compute the per entry distance d between each entry and the selection as follows:

$$distance(E, \mathbb{S}) = \frac{\sum_{i}^{|dims|} \left(\sum_{j}^{rows(\mathbb{S})} [E_i \neq \mathbb{S}_{i,j}] \cdot w_i \right) \cdot \frac{1}{rows(\mathbb{S})}}{|dims|} \tag{4}$$

In contrast to Equation 1, we transpose the input and calculate the distance per dimension instead of per entry. The distance between

one entry E and the user-defined selection of entries $\mathbb{S}$ is computed by first iterating all dimensions from $i = 1$ to the total amount of dimensions $|dims|$. Then, all rows for the *i-th* dimension are iterated from $j = 1$ to the total amount of rows (entries) $rows(\mathbb{S})$. Using Iverson Brackets [17], the *i-th* dimension of the entry is compared to all *i-th* dimensions of the selection $\mathbb{S}$ and then the average is computed. To determine the final distance value d, we compute the average value among all dimensions. This distance value, determined for each data entry individually, is inserted in a list which is sorted in descending order with respect to the distance value. Clusters of similar entries,

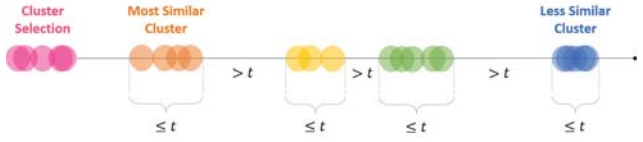

Fig. 7. One-dimensional DBSCAN algorithm for similarity values using a user-defined threshold t.

compared to the user-selection, are further derived by performing a one-dimensional density-based clustering (using the DBSCAN algorithm [13], see Figure 7). We set the threshold to 0.01 per default (applied to all examples in this paper). The threshold depends on how granular the user wants to find similar patterns. The higher the threshold, the higher the distance between found clusters is, allowing a high discrepancy within the clusters. The density-based algorithm performs as follows: Considering the threshold, the algorithm starts at the first entry of the similarity list, and successively compares the similarity value n (currently visited) to the value $n+1$. We distinguish between three cases:

- If the difference between those two values is smaller than the threshold, the entries are combined to a pattern cluster.
- If the entry of n already belongs to a pattern cluster, the entry for $n+1$ is added to the cluster.
- If the distance is greater than the defined threshold, the entry for $n+1$ starts a new pattern cluster.

Using this algorithm, we adapt to the creation of the distance matrix, which influences the outcome of the TMDS. This way, our algorithm finds patterns that already have been considered by the MDS computation. To derive the complexity of our proposed algorithm for finding similar patterns, we split the algorithm into the three elementary parts: Firstly, the complexity of computing the distance of all points to the selection lies in $\mathcal{O}(n^2)$. Secondly, the complexity of sorting all similarity values is $\mathcal{O}(n \cdot log(n))$. Thirdly, the 1D DBSCAN has the complexity $\mathcal{O}(n)$. Hence, the overall complexity of our algorithm lies in $\mathcal{O}(n^2)$ in worst case.

6 Case Study: Network Security

We demonstrate our technique in the field of network security. In the first case study in Section 6.1, we apply our technique to a real NetFlow dataset and report the gathered findings, which we discussed with our group's system administrators. To preserve privacy, we do not publish results based on data of the recent past, but use an older dataset from our university data center. The data is based on a privacy-preserving and anonymized data collection infrastructure used in previous research [15]. To validate identified patterns, we provide a ground truth based evaluation of our approach in Section 6.2, and analyze a network security dataset from the VAST Challenge 2013[1].

6.1 NetFlow Dataset: 24-Hour Network Overview

The temporal analysis of network security datasets is a highly relevant field of research. Analysis goals are suitable for our TMDS approach, because security analysts and system administrators have large datasets (e.g., NetFlow data), which reflect the ongoing connections and data flows in the underlying computer network. Within such data, the analyst can observe various attack patterns.

Previous work in network security typically focuses either on general, temporal independent, patterns (e.g., [15]), or on temporal patterns (e.g., TNV [16]), which can typically not be analyzed promptly due to scalability and level-of-detail issues. At first sight, TMDS might look similar to PortVis [34], yet our approach does fundamentally differ as discussed in Section 3. PortVis solely uses time and port range as axes to represent the events and thus particularly focuses on port scans. Our approach does not only focus on ports, but takes arbitrary (weighted) dimensions into account, and is therefore able to identify today's complex temporal attack patterns showing general behavior. Other work also uses the idea of representing sliding windows as consecutive columns [14], but focuses on providing details of heterogeneous streams, instead of providing a visualization to focus on recurring patterns. TVi [5] also operates on temporal slices using entropy, but uses PCA-based techniques to analytically identify anomalous behavior using a timeline visualization combined with histogram charts.

In this case study, we focus on all *loud* events of a full period of 24 hours of a public /16 computer network; we want to obtain a rough image of interesting events with different characteristics. To facilitate this analysis, we use Apache Spark[2] to preprocess and sample the NetFlow data files, which are about 4 to 10 GB per day, to generate a suitable CSV file of incoming data flows only. This preprocessing step reduced the network flows to 16,474 records. Focusing solely on TCP traffic leads to 6,908 records as visualized in Figure 1.

6.1.1 Data Processing

After loading the CSV file into our interactive prototype, we weight the different main dimensions with respect to increasing the impact of the IP addresses and the destination port. Because we analyze incoming network traffic, we are particularly interested in how possible attackers access services within our network. In such cases, the source port (*srcport*) is less helpful to distinguish between different attack patterns, because it is assigned by the operating system or router from a ephemeral port range, respectively. However, the destination port (*dstport*) is relevant to assign attacks to similar attack vectors. A higher weight of such ports leads to visual clusters of attacks to the same service (e.g., focusing on port TCP/80, which is the default port for HTTP traffic). After weighting the dimensions, the TMDS is computed within seconds and the visualization display is loaded.

6.1.2 Findings and Insights

It is typical for network traffic that most connections are diverse, and thus hard to distinguish – legitimate traffic behaves as expected and does not evince clear patterns or clusters. We observe the same situation in this case study. This is why many records are diversely spread on the vertical axis (light blue dots pointed out in Figure 1). However, several other interesting and unexpected patterns are salient.

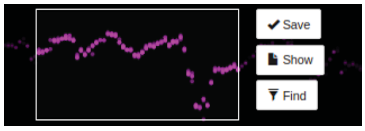

Fig. 8. Manual selection of salient patterns used to initiate the search for similar patterns (clusters). Resulting similar patterns are color coded according to their similarity ranking.

We discovered various salient visual patterns using TMDS, which are labeled from A to D in Figure 1. Finding these events with diverse characteristics without visual support of TMDS, would require the analyst to issue various manual queries on the data. Manual queries are hard to express and time-consuming without any prior knowledge.

[1] http://vacommunity.org/VAST+Challenge+2013

[2] https://spark.apache.org/

- **Pattern A: Distributed Brute-Force Attack** – A distributed bot network performs a long-term brute-force attack on port TCP/22 with the aim to break into reachable SSH servers. Using our pattern finding algorithm with an arbitrary selection of the pattern as input (see e.g. Figure 8) reveals that the attack was operated over an even longer time period. All events related to this specific attack are presented in Figure 1 and are colored magenta.
- **Pattern B: Massive Port Scan** – Drilling-down the visual pattern (highlighted in dark blue) reveals a massive port scan from a *single* external IP address to a specific exclusive set of ports (TCP/80, TCP/81, TCP/443, TCP/8000, TCP/8080) of various internal computers; the attacker is not related to the ongoing brute-force attack. The scan was operated from 10:36 until 10:56. The goal of this scan was to check for running webservers on several common ports.
- **Pattern C: Single Port Scan** – This pattern reveals a port scan to our network looking for accessible webservers on port TCP/80. In addition, some port scans search for open SMTP server on port TCP/25, which is typically performed to identify mail servers. Open mail servers can be used as open relay for sending spam.
- **Pattern D: Brute-Force Continuations** – The magenta color refers to the same characteristics as seen in Pattern A. Some attackers are still trying to attack SSH services, however in a much more subtle way than during night time as seen in Pattern A.

6.2 VAST Challenge Dataset: Identification of Events

As described in the previous Section 6.1, TMDS can be successfully applied to real network traffic. However, it is challenging to evaluate the effectiveness, because no ground truth data is available. Therefore, we apply our approach to the VAST Challenge 2013 Mini-Challenge 3 (MC3), which provides a realistic artificial dataset of a large company and a ground truth to compare with. The dataset contains several suspicious events in a computer network over a period of two weeks. Our aim is to validate if TMDS is capable of identifying notable events within this complex challenge. In contrast to the work by Chen et al. [7] who developed a highly interactive collaborative visual analysis system to address the challenge, our focus is on visually supported pattern finding. Furthermore, we focus only on the NetFlow dataset, while Chen et al. make use of all available datasets including NetFlow data, monitoring logs of a Big Brother (BB) system, and data of an intrusion prevention system (IPS).

6.2.1 Data Processing

For the VAST Challenge dataset we used similar data processing steps, as discussed in Section 6.1.1. The aim is to focus on external attacks to the company network. We filtered the available NetFlow data for incoming traffic and applied a heuristic to focus only on incoming unidirectional flows. We further filtered for all records having a source address within 10.0.0.0/8 (which reflects the whole Internet in the artificial data) and a destination address within 172.0.0.0/8, which reflects the internal company network. Additionally, we removed responses to low port numbers, which are most likely responses to outgoing connections initiated from the company network. After visualizing the data, we encountered that DoS attacks lead to vast amounts of network flow. On that score, we decided to apply an adjusted stratified sampling based on destination ports and date. We further use higher sampling rates for high-volume ports (e.g., port TCP/80 and TCP/25). In contrast to global sampling techniques, adjusted stratified sampling still enables the exposure of other subtle patterns, which otherwise are missed. We want to note, that we did not optimize the TMDS weightings for the specific attacks. We generally used $timestamp_{weight} = 0.0$ to exclude the timestamp dimension, and a weight of 1.0 for all other dimensions. The window was set to 100 and the offset to 10 entries.

6.2.2 Ground Truth Validation

Following, we present an extract of screenshots of the most salient visual TMDS patterns, which directly correlate to suspicious events and can partly be validated using the ground truth data. The TMDS patterns cover a time period of multiple days. An overview of all ground truth events is presented in Table 1.

Event ID	Subtlety	Event Type	Data Source	TMDS	Pattern
(1)	Questions only	Videoconference	-	-	-
(2)	Questions only	Threatening Letter	-	-	-
(3)	Subtle	Port Scans	NetFlow/BB	✓	Fig. 9
(4)	Subtle	Port Scans	NetFlow	✓	Fig. 9
(5)	Obvious	DoS	NetFlow	✓	Fig. 10
(6a)	Subtle	Server Crash	NetFlow/BB	×	-
(6b)	Subtle	Server Return	NetFlow	(×)	-
(7)	Subtle	Port Scans	NetFlow	✓	Fig. 10
(8a)	Obvious	DoS	NetFlow/BB	✓	Fig. 11
(8b)	Obvious	DoS	NetFlow	(✓)	Fig. 11
(9a)	Subtle	Server Crash	NetFlow/BB	×	-
(9b)	Subtle	Server Return	NetFlow	(×)	-
(10)	Subtle	Malicious Redirects	NetFlow	×	-
(11)	Obvious	Exfiltration	NetFlow	-	-
(12)	Obvious	Port Scans	NetFlow	✓	Fig. 12
(13)	Obvious	Port Scans	NetFlow	✓	Fig. 12
(14)	Obvious	Exfiltration	NetFlow	-	-
(15)	Questions only	Threatening Letter	-	-	-
(16)	Obvious	Network Down[3]	NetFlow	✓	-
(17)	Obvious	Port Scans	NetFlow/IPS	✓	Fig. 13
(18)	Obvious	Port Scans	NetFlow/IPS	✓	Fig. 14
(19)	Obvious	Failed DoS	NetFlow/IPS	✓	Fig. 14
(20)	Obvious	Failed Exfiltration	IPS	-	-
(21)	Obvious	Port Scans	NetFlow/IPS	✓	Fig. 14
(22)	Subtle	Botnet Infection	NetFlow	-	-
(23)	Obvious	Botnet Communication	NetFlow	-	-
(24)	Obvious	Port Scans	NetFlow/IPS	✓	-
(25)	Obvious	Port Scans	NetFlow/IPS	✓	-
(26)	Obvious	Botnet DoS Attacks	NetFlow/IPS	-	-
(27)	Obvious	Botnet DoS Attacks	NetFlow/IPS	-	-
(28)	Obvious	Port Scans	NetFlow/IPS	✓	-
(29)	Obvious	Port Scans	NetFlow/IPS	✓	-

Table 1. The ground truth for the VAST Challenge 2013 MC3 consists of 29 official events. After analyzing the data with default weightings, we compared our findings with the official ground truth and used a check mark to highlight successfully identified event patterns using TMDS.

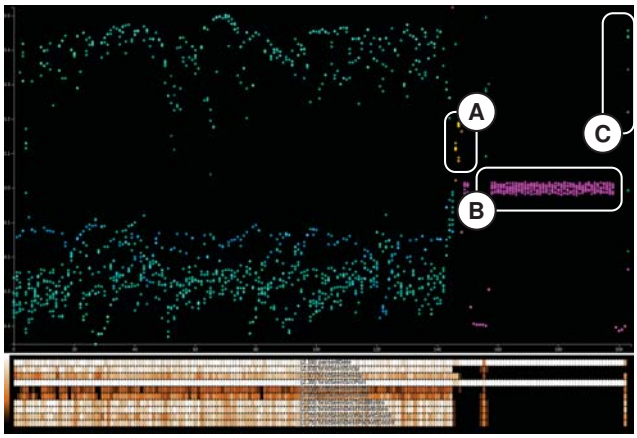

Fig. 9. TMDS for the first day on 2013-04-01 00:00 to 23:59 revealing various visual patterns related to Event (3) and (4) in Table 1.

From the given data it is not possible to identify Event (1) and (2), because they were not visible in the data at all. The organizers of the VAST Challenge provided the option for participants to ask specific questions, which would have revealed more details to such events. The first official identified event in the data is Event (3), which is classified by the data provider as *subtle* event. Using TMDS, Figure 9 points out this event as Pattern A and B. The diverse blue and green colored patterns on the top-left and bottom-left relate to normal legitimate incoming network traffic. Using details on demand, we are able to inspect the underlying flow data of the patterns and can judge and

[3]The company takes the network down to investigate security concerns and to install an intrusion prevention system (IPS).

classify them. Pattern A is indeed suspicious and relates to an attack from source IP 10.6.6.6 to 172.30.0.x machines, which qualifies according to the ground truth "as subtle because firewall allows mainly ports 25 and 80". Pattern B is a result of Event 4 and is described as "high volume web browsing traffic". The diversity matrix below the TMDS plot highlights correlations (black rectangles) between source IP, destination IP, and destination port using the Shannon Entropy. Due to the low entropy, we see that the attacker successively generated almost identical requests. The pattern is "followed by portscans", which is made visible as subtle Pattern C on 2013-04-01 22:18.

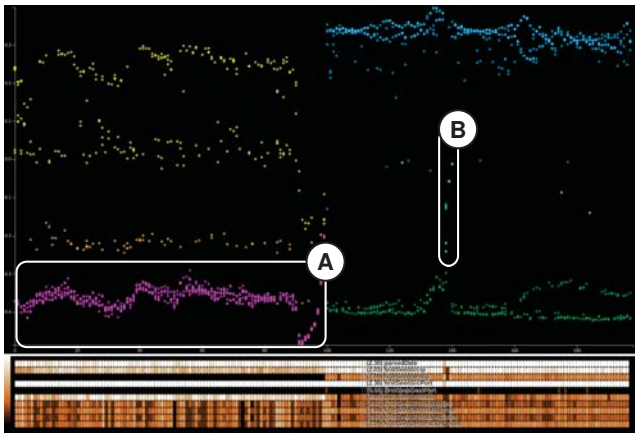

Fig. 10. TMDS for the 2nd day on 2013-04-02 00:00 to 23:59 highlights an obvious DoS attack (Pattern A) and a subtle port scan (Pattern B).

Figure 10 shows the TMDS plot for the second day. The salient pattern A is a DoS attack lasting from 05:22 to 07:22 and originates from 10 attackers to webserver 172.30.0.4. According to the ground truth, this webserver becomes temporarily unresponsive (Event 6a), which cannot be seen by TMDS plots, because we focus our analysis on incoming traffic only and do not highlight missing data. Additionally, we do not integrate the Big Brother (BB) system monitoring dataset, which would have identified this event. Event 7, related to subtle port scans from 10.6.6.6 and 10.7.7.10 attacking primarily port TCP/25, is displayed as Pattern B in Figure 10. The patterns become clearly visible by cluster colors after selecting a part of Pattern A as reference.

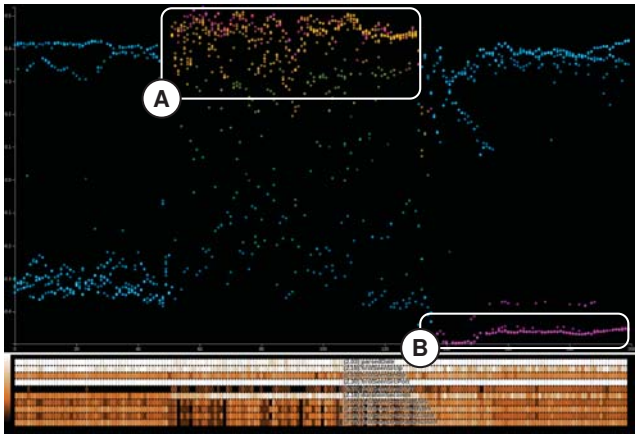

Fig. 11. TMDS for the 3rd day on 2013-04-03 00:00 to 23:59 with sudden pattern change (A) related to an ongoing distributed DoS attack and another attacker (B) attacking primarily another webserver.

On the 3rd day, Figure 11 distinctly shows a major pattern change from 9:30 until around 11:48, visible as Pattern A and related to another ongoing distributed DoS attack. We note that the pattern is not homogeneous with respect to the found clusters (shown as various colors). The pattern is dominated by the orange cluster, which actually represents Event 8a in the ground truth (a distributed DoS originating from various attackers). In contrast, the magenta colored dots, which continue in Pattern B, relate to Event 8b; attacker 10.15.7.85 primarily attacks a different webserver (172.20.0.15) compared to the others.

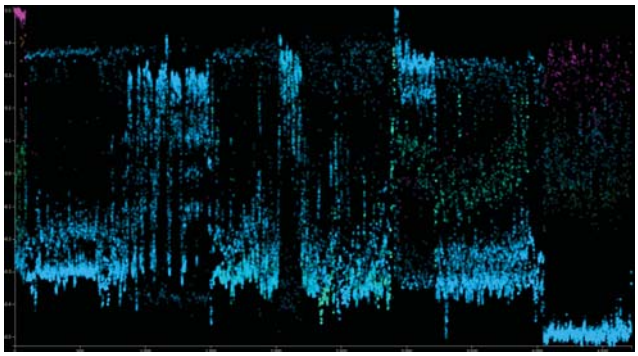

Fig. 12. TMDS for the 6th day on 2013-04-06 00:00 to 23:59 revealing unexpected diverse port scanning patterns. The firewall seems to be not working anymore, because heavy port scans on arbitrary ports do reach the company network.

On the 6th day, the TMDS reveals a complete change of the network behavior. Until now, the firewall seemed to successfully block most of the unknown destination ports from external access. However, while exploring the patterns on the sixth day in Figure 12, it becomes immediately apparent that the various patterns result from flows to arbitrary destination ports and thus lead to less defined clusters. This observation matches the description in the ground truth: An administrator computer got infected (which cannot be seen in the data at all) and an attacker decided "to change firewall settings, opening all ports" in the company network. The different port scans are visible in the TMDS plot, however the exfiltrations (Events 11 and 14) are only visible, if we would consider outgoing traffic. The display is almost completely cluttered by heavy port scans (blue and green colored clusters), which originate from 10.9.81.5 and 10.10.11.15 (Events 12 and 13).

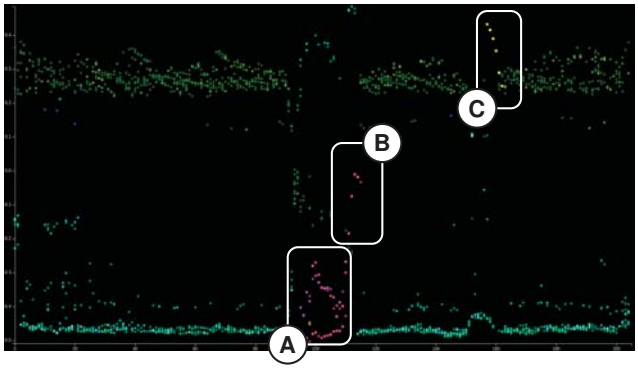

Fig. 13. TMDS for the 10th day on 2013-04-10 00:00 to 23:59 highlights three port scans, which are summarized as Event 17 in the ground truth.

From the overall challenge description, we know that the company installed an intrusion prevention system. Based on the TMDS plots of the 10th day, as depicted in Figure 13, the network traffic seems to be back to normal operation. However, the plot also reveals entropy changes starting around 12:20, which reveal three interesting clusters. Pattern A and B can be identified again as portscans from attacker 10.138.235.111 and 10.6.6.7, while Pattern C is mostly dominated by 10.13.77.49. All three patterns directly relate to Event 17.

The patterns on 2013-04-11, as depicted in Figure 14, are very diverse. Especially the light gray cluster broadly spreads from around 11:55 until 12:57 revealing a distributed DoS attack (Event 19). The

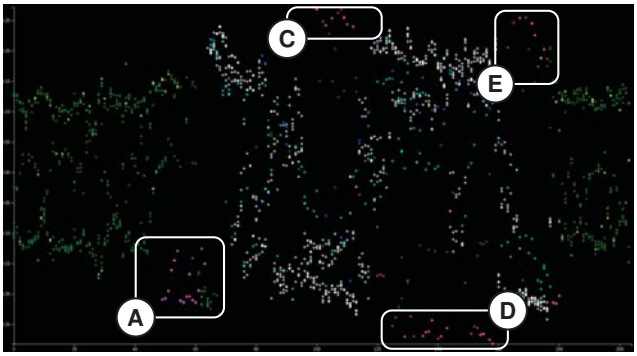

Fig. 14. TMDS for the 11th day on 2013-04-11 00:00 to 23:59 highlighting three distinctive port scans. The scattered gray-colored dots from 11:55 until 12:57 reveal a diverse distributed DoS attack (Event 19).

attack seems to facilitate a high volume of connections and the ground truth confirms that attackers use "a mix of duration and payload sizes", thus leading to more diverse and less clear patterns. A port scan from 10.12.15.152 between 10:35 and 11:21 is visible as Pattern A and is colored magenta (Event 18). Similarity clustering and respective color assignment reveal that Pattern C, D, and E actually belong to the same long-term port scan from a single attacker 10.6.6.7 (Event 21).

6.2.3 Summary and Limitations

As summarized in Table 1, the ground truth contains a total amount of 29 events, while three events (6, 8, 9) consist of two coherent sub events. Three other events (1, 2, 15) are not data-dependent, because they are part of questions, which are not part of the data. Event 20 is only visible in IPS data. We focused on *incoming* NetFlow data only, while six events (11, 14, 22, 23, 26, 27) do relate to outgoing network traffic, because the suspicious traffic comes from internal machines.

This leaves us with a total of 19 events, which we aimed to detect with TMDS within our case study. We successfully identified the distinctive patterns of 16 events. Overall, we only missed three events (6, 9, 10), and hence identified more than 84% of applicable ground truth events. Besides that, we were able to identify additional suspicious events, which we did not report, because even they were interesting from a system administrator's perspective – they were not part of the official ground truth data and hence could not be validated for certain.

TMDS was able to reveal interesting patterns, which actually corresponded to suspicious events verified by the ground truth. However, compared to the work of Chen et al. [7], our system provides a general approach for the analysis of multivariate data and therefore does not provide additional correlated views tailored to the needs of security analysts. Interestingly, we were able to detect most of the patterns, although further manual analysis (details on demand) of the underlying data of identified patterns was needed to finally judge and classify the event. For example, it is not directly visible in TMDS which hosts are attacked. To identify actual hosts, a manual selection of the pattern is needed, which retrieves a list of underlying NetFlow records. At this step further visualizations (e.g., Parallel Coordinates) can be integrated to summarize records belonging to an identified suspicious pattern. Even without integrated views, TMDS yet provides effective identification rates for valid patterns. Consequently, the integration of TMDS in security applications seems to be promising and can improve various visual analytics applications.

7 Extensions and Discussion

TMDS is geared to the visual analytics process [28] and enables a novel analysis of temporal multivariate data. We demonstrated in Section 6 that TMDS is visually able to make temporal and sequential patterns salient, involving domain knowledge and interaction. As pointed out, the application of TMDS to network data works fine, and we retrieve plausible results using rules of thumb. Yet we cannot provide fixed parameters (e.g., window size, step size), because it depends on the data characteristics and size. One way to suggest plausible parameters, is to generate multiple plots, taking into account the window and overlap size discussion of Section 4.1. Afterwards, apply visual quality metrics to the plots such as Hough transform or contour tracking.

We note that for pragmatic reasons and as a first step, for the MDS projection we chose a rectangular windowing function for all data entries contained in a sliding window. As our sliding window in practice spans a larger number of entries, the changes introduced by the unweighted exit and entry of entries on each sliding step do each not have a huge impact on the projection result. However, we expect that for smaller window sizes and/or larger offsets, we would require a non-uniform weighting scheme to provide sufficient stability of the projections. E.g., Gaussian or triangular weighting schemes centered on the sliding window may be useful [11]. We tested with different parameters, finding that with an offset of circa 10% of the window size, and window size of at least tens of entries, we achieve sufficiently stable results for unit weighting. We note we leave assessment of the effect of alternative weighting schemes in respect to window size, offset, and data and analysis tasks as an important subject for future work.

Another possible extension to our approach represents the additional visualization of interesting dimensions. Suppose, several hundred dimensions are taken into account. Finding automatically interesting or relevant dimensions and plot them on top of the visualization is challenging. We therefore recommend to make use of the proposed diversity matrix and apply ordering heuristics for certain use cases.

In addition, the application to categorical data is only preliminary and can be extended in various ways. For instance, involving the user [22] and adding semantic information like hierarchies, etc., in the categories, can improve results and enhance the analysis process.

8 Concluding Remarks and Perspectives

We presented a novel approach to identify patterns in multivariate data evolving over time. We introduced TMDS as a temporally smoothed, time-dependent 1D MDS plot of multivariate data. The plot allows to explore for data areas of interest, based on evolving structures in the temporal 1D MDS plot. A linked heatmap shows attribute diversity and allows to compare the global MDS patterns for properties of the underlying attributes, supporting the analysis in detail. The use of a sliding window enables fast parallel computation, but also smooths the MDS result by preventing abrupt changes between entries based on the similarity value. In combination with the introduced diversity matrix, correlations between dimensions can be efficiently spotted. This way, the system supports conclusions drawn from visualized patterns.

Dealing with multivariate or high-dimensional data is a difficult problem in general. TMDS can be applied to data with various dimensions. However, if TMDS does not reveal salient patterns due to the vast amount of dimensions, it is up to the user to restrict the analysis to a domain specific set, which can for example be efficiently achieved using the suggestion functionality. For the identification of patterns, we proposed a density based algorithm following the computation of the distance matrix. It enables the segmentation of visualization with respect to similar patterns that evolve over time. We furthermore showed the usefulness of our approach in a network security case study, which considered a real dataset as well as a ground-truth dataset provided by the VAST Challenge 2013.

Our sliding window approach provides an initial size for first results but requires refinement by the user in order to make patterns salient. In future work, we plan to evaluate the sliding window to provide a rule for different datasets as well as tasks and applications. We will also focus on improvements regarding real-time processing and scalability in the domain of network security, so that TMDS can be used in a big data environment providing results for incoming new data.

Acknowledgments

We are thankful to Ming Hao and Wei-Nchih Lee of Hewlett-Packard Labs for fruitful discussions on multivariate data analysis and an earlier instance of the solution. This work was partly supported by the EU project Visual Analytics for Sense-making in Criminal Intelligence Analysis (VALCRI) under grant number FP7-SEC-2013-608142.

REFERENCES

[1] W. Aigner, S. Miksch, H. Schumann, and C. Tominski. *Visualization of Time-Oriented Data*. Human-Computer Interaction Series. Springer, 2011.

[2] Algorithmics Group. *MDSJ: Java Library for Multidimensional Scaling (Version 0.2)*. University of Konstanz, 2009. Available at `http://www.inf.uni-konstanz.de/algo/software/mdsj/`.

[3] D. F. Andrews. Plots of high-dimensional data. *Biometrics*, 28(1):pp. 125–136, 1972.

[4] J. Bernard, N. Wilhelm, M. Scherer, T. May, and T. Schreck. Timeseries-paths: Projection-based explorative analysis of multivarate time series data. *Journal of WSCG*, 20(2):97–106, 2012.

[5] A. Boschetti, L. Salgarelli, C. Muelder, and K.-L. Ma. Tvi: A visual querying system for network monitoring and anomaly detection. In *Proceedings of the 8th International Symposium on Visualization for Cyber Security*, VizSec '11, pages 1:1–1:10, New York, NY, USA, 2011. ACM.

[6] J. Chambers. *Graphical methods for data analysis*. Chapman & Hall statistics series. Wadsworth International Group, 1983.

[7] S. Chen, C. Guo, X. Yuan, F. Merkle, H. Schaefer, and T. Ertl. Oceans: Online collaborative explorative analysis on network security. In *Proceedings of the Eleventh Workshop on Visualization for Cyber Security*, VizSec '14, pages 1–8, New York, NY, USA, 2014. ACM.

[8] H. Chernoff. The use of faces to represent points in k-dimensional space graphically. *Journal of the American Statistical Association*, 68(342):pp. 361–368, 1973.

[9] T. Cox and A. Cox. *Multidimensional Scaling, Second Edition*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. CRC Press, 2000.

[10] T. Crnovrsanin, C. Muelder, C. D. Correa, and K. Ma. Proximity-based visualization of movement trace data. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology, IEEE VAST 2009, Atlantic City, New Jersey, USA, 11-16 October 2009, part of VisWeek 2009*, pages 11–18, 2009.

[11] P. S. R. Diniz, E. A. B. da Silve, and S. L. Netto. *Digital Signal Processing: System Analysis and Design*. E-Libro. Cambridge University Press, 2010.

[12] T. Dwyer and D. R. Gallagher. Visualising changes in fund manager holdings in two and a half-dimensions. *Information Visualization*, 3(4):227–244, 2004.

[13] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*, pages 226–231, 1996.

[14] F. Fischer and D. A. Keim. NStreamAware: Real-time visual analytics for data streams to enhance situational awareness. In *Proceedings of the Eleventh Workshop on Visualization for Cyber Security*, VizSec '14, pages 65–72, New York, NY, USA, 2014. ACM.

[15] F. Fischer, F. Mansmann, D. A. Keim, S. Pietzko, and M. Waldvogel. Large-Scale Network Monitoring for Visual Analysis of Attacks. In J. R. Goodall, G. Conti, and K.-L. Ma, editors, *Visualization for Computer Security*, number 5210 in Lecture Notes in Computer Science, pages 111–118. Springer Berlin Heidelberg, 2008.

[16] J. Goodall, W. Lutters, P. Rheingans, and A. Komlodi. Preserving the big picture: visual network traffic analysis with TNV. In *IEEE Workshop on Visualization for Computer Security, 2005. (VizSEC 05)*, pages 47–54, Oct. 2005.

[17] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 1994.

[18] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kauffman, 2nd edition, 2006.

[19] A. Hinneburg, C. C. Aggarwal, and D. A. Keim. What is the nearest neighbor in high dimensional spaces? In *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 506–515, 2000.

[20] Y. Hu, S. Wu, S. Xia, J. Fu, and W. Chen. Motion track: Visualizing variations of human motion data. In *IEEE Pacific Visualization Symposium PacificVis 2010, Taipei, Taiwan, March 2-5, 2010*, pages 153–160, 2010.

[21] A. Inselberg and B. Dimsdale. Parallel coordinates: A tool for visualizing multi-dimensional geometry. In *IEEE Visualization*, pages 361–378, 1990.

[22] S. Johansson. Visual exploration of categorical and mixed data sets. In *Proceedings of the ACM SIGKDD Workshop on Visual Analytics and Knowledge Discovery: Integrating Automated Analysis with Interactive Exploration, Paris, France, June 28, 2009*, pages 21–29, 2009.

[23] I. Jolliffe. *Principal component analysis*. Springer series in statistics. Springer-Verlang, 1986.

[24] E. Kandogan. Star coordinates: A multi-dimensional visualization technique with uniform treatment of dimensions. In *In Proceedings of the IEEE Information Visualization Symposium, Late Breaking Hot Topics*, pages 9–12, 2000.

[25] J. Kehrer and H. Hauser. Visualization and visual analysis of multifaceted scientific data: A survey. *IEEE Trans. Vis. Comput. Graph.*, 19(3):495–513, 2013.

[26] D. A. Keim, M. Ankerst, and H. Kriegel. Recursive pattern: A technique for visualizing very large amounts of data. In *IEEE Visualization*, pages 279–286, 1995.

[27] D. A. Keim, M. C. Hao, U. Dayal, and M. Hsu. Pixel bar charts: a visualization technique for very large multi-attribute data sets? *Information Visualization*, 1(1):20–34, 2002.

[28] D. A. Keim, F. Mansmann, J. Schneidewind, and H. Ziegler. Challenges in Visual Data Analysis. In *Information Visualization (IV 2006)*. IEEE, IEEE Press, 2006.

[29] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, 1982.

[30] J. LeBlanc, M. O. Ward, and N. Wittels. Exploring n-dimensional databases. In *IEEE Visualization*, pages 230–237, 1990.

[31] H. Liu and H. Motoda, editors. *Computational Methods of Feature Selection*. Chapman and Hall/CRC, Boca Raton, Oct. 2007.

[32] B. Manly. *Multivariate Statistical Methods: A Primer, Third Edition*. Taylor & Francis, 2004.

[33] Y. Mao, J. V. Dillon, and G. Lebanon. Sequential document visualization. *IEEE Trans. Vis. Comput. Graph.*, 13(6):1208–1215, 2007.

[34] J. McPherson, K.-L. Ma, P. Krystosk, T. Bartoletti, and M. Christensen. PortVis: A Tool for Port-based Detection of Security Events. In *Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, VizSEC/DMSEC '04, pages 73–81, New York, NY, USA, 2004. ACM.

[35] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. Adaptive computation and machine learning series. MIT Press, 2012.

[36] C. Shannon. A mathematical theory of communication. *Bell System Technical Journal, The*, 27(3):379–423, July 1948.

[37] E. H. Simpson. Measurement of diversity. *Nature*, 1949.

[38] A. Tatu, H. Theisel, T. Braunschweig, M. Magnor, T. Braunschweig, M. Eisemann, T. Braunschweig, D. Keim, and J. Schneidewind. Combining automated analysis and visualization techniques for effective exploration of high dimensional data. In *IEEE Symposium on Visual Analytics Science and Technology*, 2009.

[39] E. Tufte. *Envisioning Information*. Graphics Press, Cheshire, CT, USA, 1990.

[40] M. O. Ward and Z. Guo. Visual exploration of time-series data with shape space projections. *Comput. Graph. Forum*, 30(3):701–710, 2011.

[41] T. Yang, J. Liu, L. McMillan, and W. Wang. A fast approximation to multidimensional scaling. In *IEEE workshop on Computation Intensive Methods for Computer Vision*, 2006.