PieceStack: Toward Better Understanding of Stacked Graphs

Tongshuang Wu, Yingcai Wu, *Member, IEEE,* Conglei Shi, *Member, IEEE,*
Huamin Qu, *Member, IEEE,* and Weiwei Cui, *Member, IEEE*

Abstract—Stacked graphs have been widely adopted in various fields, because they are capable of hierarchically visualizing a set of temporal sequences as well as their aggregation. However, because of visual illusion issues, connections between overly-detailed individual layers and overly-generalized aggregation are intercepted. Consequently, information in this area has yet to be fully excavated. Thus, we present PieceStack in this paper, to reveal the relevance of stacked graphs in understanding intrinsic details of their displayed shapes. This new visual analytic design interprets the ways through which aggregations are generated with individual layers by interactively splitting and re-constructing the stacked graphs. A clustering algorithm is designed to partition stacked graphs into sub-aggregated pieces based on trend similarities of layers. We then visualize the pieces with augmented encoding to help analysts decompose and explore the graphs with respect to their interests. Case studies and a user study are conducted to demonstrate the usefulness of our technique in understanding the formation of stacked graphs.

Index Terms—Statistical graphics, Time series visualization, Stacked graphs

1 Introduction

Stacked graphs, which are among of the oldest and the most fundamental visual representations of multiple time series data [26], have been widely adopted in various applications (e.g., email messages [14], sentiments [5], music listening histories [4], [36], etc.). Although this combined visualization of individual and aggregated values has revealed interesting stories in these fields, there are tasks where analysts may find stacked graphs insufficient.

An important task is understanding the causality between the overall aggregated shape and individual layers. For example, a social media analyst may use a stacked graph to explore the popularity change in a set of Twitter hashtags over time. He may locate a pattern of interest, such as a dent in the overall shape, and may want to understand which hashtag(s) causes the bump and how. However, many different ways could have led to the bump. It may have resulted from a majority of the hashtags with similar small bumps, or a few of the hashtags while others stay stable or even experience a dent that actually has a negative contribution to the aggregation. Finding the exact composition of a pattern is important, because different compositions may provide different clues and eventually lead to different insights.

The aforementioned task and various commonly seen similar tasks all essentially probe the connection between aggregation and individual layers. Unfortunately, traditional stacked graphs are not effective for those tasks because of several limitations in their expressiveness, which makes understanding complex constructions non-trivial (Fig. 1). First, the comparability for stacked graphs is weak. The baselines of layers are neither parallel nor overlapping, which makes stacked graphs inefficient for tasks that involve comparing the trends or shapes of layers. Second, the layer order is consistent during the entire period. While consistent order helps users track individual layers, it also tends to emphasize global features and neglect local ones. For example, placing layers with similar trends together can help users discover the major trends effectively. However, since the similarity relationships between layers may be totally different over time, finding a universally good order for all time points is nearly impossible, which complicates the analysis of causalities at multiple time points. Third, stacked graphs have scalability issues. The number of layers in a stacked graph could soon lead to visual clutter, which makes the comparison more difficult.

This study addresses these challenges by proposing PieceStack , a new visual analytic design that enables improved interpretation for stacked graphs. We first design a new division-and-group clustering algorithm to deal with complex temporal data flexibly. Based on the clustering results, several visual elements are designed and integrated into PieceStack to reveal the relationships between layers and the overall shape, and further to help users explore the construction of the overall shape. In addition, we allow a series of interactions in our system for users to refine the results based on their interests. Our main contributions are described as follows:

- This study is the first to systematically examine how stacked graphs are generated with the individual layers by exploring similarities/differences between layers, as well as between layers and the overall trend, especially among local intervals.
- We propose a new algorithm of partial sequence clustering on stacked graphs, which interprets every time period in detail without being constrained by the overall complex changes.
- Our design of an interactive system, PieceStack, empowers analysts to better understand the formation of stacked graphs.

- *Tongshuang Wu and Huamin Qu are with the Hong Kong University of Science and Technology. This work was done when Tongshuang Wu was an intern at Microsoft Research. E-mail: {twuac, huamin}@ust.hk.*
- *Yingcai Wu is with State Key Lab of CAD & CG, Zhejiang University. E-mail: ycwu@cad.zju.edu.cn.*
- *Conglei Shi is with the IBM T.J, Watson Research Center, Yorktown Height, NY. E-mail: conglei.shi@us.ibm.com.*
- *Weiwei Cui is with Microsoft Research and is the corresponding author. E-mail: weiwei.cui@microsoft.com.*

Manuscript received October 3, 2015; revised January 4, 2015.

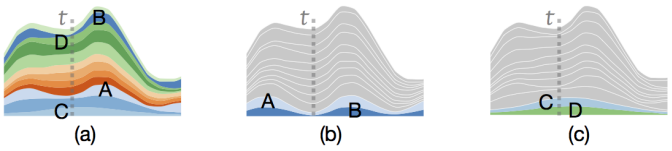

Fig. 1. An aggregated dent at t and its involved layers contributing differently: (a) the original graph, (b) similar layers A and B contributing positively, (c) similar layers C and D, in contrast, contributing negatively. Layers in (b) and (c) are reordered to emphasize the otherwise inapparent similarities of layers and the negative contributions in (c).

2 Related Work

2.1 Stacked Graph

A stacked graph is a well-established visualization method for time-varying data on one common timeline. Many recent studies have focused on improving its visual perception. The first major step was made by Havre et al. [15], who proposed ThemeRiver in which layers are visualized with smooth curves and stacked in a symmetrical shape. Given improved visual pleasantness, their idea was widely adopted and inspired more research. For example, Byron and Wattenberg [4] introduced Streamgraphs, which improved the legibility and aesthetics of stacked graphs. The color was also used to make the hierarchical structure more visible in the stacked graphs [33].

Besides improving aesthetics and legibility, researchers have attempted to improve utilities by combining stacked graphs with additional tools and visual elements. For example, NameVoyager [16] combined filtering capability with stacked graphs, thereby enabling rapid exploration based on prefix text search. Baur et al. [2] presented TouchWave as an extension of stacked graphs for multi-touch capable devices that provides a variety of layout adjustments to address the problems of legibility, comparison, and scalability. RankExplorer [28] embedded color bars and transition glyphs to help users analyze ranking changes. TextFlow [7] enhanced stacked graphs with a flow-based metaphor to support merging and splitting relationships among layers. TIARA [21] integrated text summarization techniques to explore large collections of documents. Dork et al. [8] described a highly interactive system based on tailored stacked graphs to visualize a continuously updating information stream. LoyalTracker [29] was designed as an augmented stacked graph to show the flows that enter or leave the graph so that user loyalty can be tracked for search engines. EvoRiver [31] separated stacked graphs into different strips to deal with time-varying coopetition power. By augmenting the richness of visual encodings and incorporating various interactions, these studies have equipped stacked graphs with new capabilities to deal with various types of data or to support specific needs from applications. Our work also belongs to this category. However, compared with the existing work, ours is more introspective by focusing on helping people understand the construction of stacked graphs.

2.2 Clustering Algorithm

Clustering of time series data is useful in multiple domains [27]. A comprehensive investigation can be seen in [20]. For instance, Singhal et al. [30] clustered multivariate time-series data based on principal component analysis (PCA) and Mahalanobis distance, Wang et al. [34] also processed measured distances with the structural statistical characteristics for clustering. Xiong et al. [35] proposed a model-based procedure, clustering time series with autoregressive moving average, and Fröhwirth-Schnatter et al. [10] pooled time series into several groups using finite-mixture models.

Trajectory clustering and curve clustering, which are characterized by high dimensionality and the need to preserve the shape-level smoothness or temporal information, could also be used for time series clustering essentially. Gaffney [11] proposed a clustering model based on polynomial mixture model with expectation-maximization (EM) algorithm, and [1] fitted the temporal data with B-spline before the clustering procedure to emphasize the functional nature of the objects. Several strengths and weaknesses of the trajectory clustering procedure are examined in [22].

Although these algorithms have been proved useful in many fields, they all cluster time series sequences, trajectories, or curves as a whole, which is not useful in discovering partial similarities. Time series clustering involves a popular branch that breaks away from the whole period, namely, sub-sequence time series clustering (STS) [12], [13], [18]. STS clusters sub-sequences extracted from a given single time series. However, STS could not be simply adjusted for our objective because it predefines fixed features with a clipping window as inputs for comparison, while the feature space for partial sequence comparison could vary significantly. The most capable algorithm for catching common sub-trajectories is TRACLUS [19], which partitions a trajectory into a set of line segments, and then groups similar segments together. The visual navigation of the simulation processes' parameter space in [3] works similarly, in the sense that it partitions time-dependent volume data and then processes density-based clustering. However, both algorithms fail to consider the temporal synchronism of the data: the former one only focuses on the possible spatial paths of the trajectories, and the latter collects all the segments as one set for further clustering without considering time period correspondence. Inspired by their partition-and-group frameworks, we propose a clustering algorithm (Section 5) specifically to discover partial trends in stacked graphs.

3 Design Consideration

3.1 Problem Analysis

In previous research, we incorporated stacked graphs into several designs to analyze various data. we found users were easily attracted to shape-related features, such as bumps, dents, or even waves in the overall aggregated shape, and were naturally curious about their causes. Their related concerns could be categorized into three progressive groups, namely:

Layer versus layer. This category refers to basic questions related to comparing layers. Questions like finding layers' similar trends in a short period are basic and important for more advanced patterns, such as their relations to the overall shape. However, since scalability and comparability are short tabs for stacked graphs, finding similar layers usually requires additional tools. Moreover, visual clutter also makes it difficult to directly find significant patterns from each layer.

Layer versus aggregation. Such questions concern the relations between individual layers and their aggregation, with the core interest lying in the generation of the latter. Some interactions like layer reordering and baseline straightening can help with these tasks. However, they could soon become inadequate when patterns become more complicated or involve multiple time points. Thus, more advanced techniques are required.

Aggregation versus aggregation. This set of questions is more about understanding higher-level aggregated features. Be-

sides understanding one aggregated pattern, connecting the causes of different patterns is also important: it can further help users gain insights into the structure of the aggregation, which is the dominant visual factor. In addition, users may find that the overall shape loses too many details. Therefore, spliting the stacked graph into two or more graphs with more salient features could provide more concrete insights.

3.2 Requirement Analysis

In response to the aforementioned questions, we compile the following list of design requirements to further explore the relations among layers and their aggregation.

Partial clustering. Supporting clustering is the basic and foremost requirement of our system. Clustering layers with similar trends is an obvious method to reduce difficulty in problems involving comparisons. Because of the frequent changes in the data, partial clustering is further desired to interpret a stacked graph correctly, such that the local features could be emphasized.

Showing clustering results in the stacked graph. Clustering clips the stacked graph into multiple groups of sub-layers and reduces the overhead of cognition. Therefore, clusters should be encoded into the original stacked graph intuitively (e.g., the durability of a cluster), and the construction of the total stacked graph should be indicated with all the clusters.

Tracking the distributions of layers in the entire period. Partial clustering may segment a layer into multiple clusters based on its trends at different time points. Tracking how a layer groups with other layers over time may reveal the frequency by which the layer breaks away from its previous group. The clustering distribution of a layer could also reveal its overall development as well as its behavioral similarities and differences with other layers.

Comparing clusters at certain time points. Although clustering could roughly reveal the trend distributions of layers, users have difficulty in understanding the formation of aggregations visually. Do all clusters share similar trends with the aggregation, or does it merely convey the patterns of clusters with numerically significant sequences? To understand this, we have to encode the contributions of clusters to the overall shape. Besides, aggregated features at different time points may be caused by different groups of layers. Understanding the relations between the causes can further help users understand the evolution of layers.

Filtering region of interests. Although the burden of clutter is lightened by clustering, stacked graphs still suffer from visual ambiguity because of biased baselines. Therefore, filtering is needed so that some regions of interest could be extracted and studied individually or comparatively with the filtered out ones.

Supporting interactive refinements. Though the system aims to automatically interpret stacked graphs, it should still implement interactions to help explore clustering and display results.

4 System Overview

The objective of PieceStack is to help understand the relations among individual layers and those with the overall aggregated shape of stacked graphs. It is possible to encode statistical measurements (correlations, similarities, etc.) individually in a supportive view. For instance, a density map could express the similarities of layers and the aggregated shape at every time point, with dark red being highly similar (e.g., bump versus bump) and dark green highly dissimilar. However, imagine if various thick layers are ordered consecutively, the reds and greens may interlace too frequently for making effective comparisons between layers, or to observe if aggregations at different points are constructed in the same way. Similar ideas projecting measurements into other forms (e.g., scatter plots) could experience similar difficulties: given that the spatial information (e.g., temporal sequences, layer ordering, width, etc.) could not be mapped precisely onto the supportive views, the switch of attentions could introduce difficulties into observations of specific layers or temporal positions. Therefore, we focus primarily on directly augmenting stacked graphs, such that the additional encodings be seamlessly integrated without loss of stacked graphs' original information.

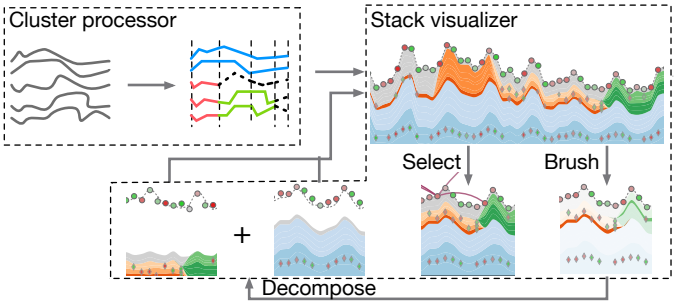

Fig. 2. System overview: PieceStack consists of two major modules, namely, clustering processor and stack visualizer.

Fig. 2 summarizes our design with its two main modules. When a collection of temporal sequences (essentially layers in traditional stacked graphs) is loaded to PieceStack, sequences are first partitioned and clustered based on their local trend similarities with the clustering processor. The stack visualizer then transforms the output (i.e., a set of clusters of sequence segments, each with respect to a certain time interval) into a comprehensible overview visualization. The overview is built upon a cluster-based stacked graph presenting cluster characteristics, with overlaid glyphs that explain the contributing factor of the clusters at each time point. Given the overview information, users could interact with PieceStack, such as brushing, selecting, and decomposing it, to further explore the effects of layers on the aggregation and the correlations between aggregations at different time points.

5 Partial Time Series Clustering

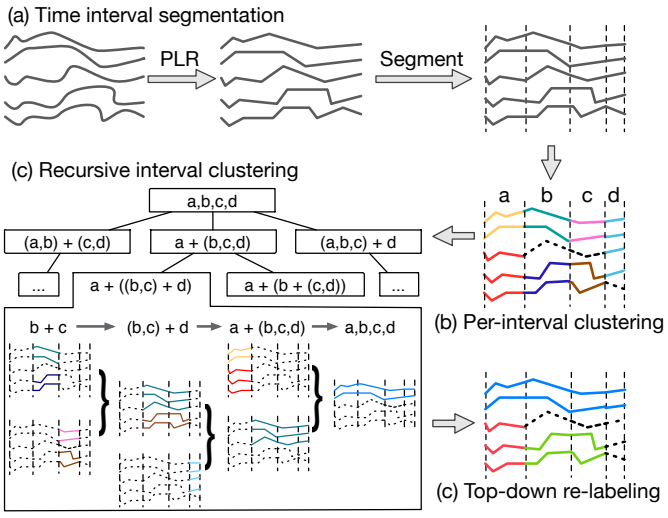

Fig. 3. Clustering algorithm workflow and the four steps, namely, (a) time series segmentation, (b) per-segment clustering, (c) recursive cluster merging, and (d) top-down re-labeling.

Since layers' similarities could vary greatly in different time intervals, simply clustering time sequences on the whole time

period could miss local common sub-sequences. Therefore, our clustering algorithm aims at partial trend similarities which, as the name indicates, refers to similar shapes in curves defined on the basis of some specific time intervals. To achieve this, we borrow the idea of partition-and-grouping from [19] to find dynamic patterns of varying lengths. Strategically, our algorithm starts with time interval segmentation (Fig. 3(a)), which simplifies and over-segments the whole time period into small intervals with varying lengths, so to reveal local similarities properly with layer segments defined on these intervals. We analyze the in-interval and cross-interval similarities with these segments thereafter: we first group similar segments in each time interval independently to locate patterns in static intervals (Fig. 3(b)). Then, by comparing the grouping results in different intervals, we construct the ones of varying lengths that best represent the evolution phases of the dataset (Fig. 3(c) and (d)). Because of the partition-and-grouping involved, the clustering strategy is not exclusive. Therefore, we identify the following criteria in designing the algorithm:

C.1 At each time point, the most general similarities should be found and the understandability be preserved by preventing the splitting of the stacked graph into too many clusters.

C.2 For the entire period, similar patterns with long duration should be found to reveal stable and disciplinary patterns instead of fragmentary ones.

The steps involved, namely Fig. 3(a) time interval segmentation which finds segments representing original layers, (b) per-interval clustering which groups similar segments locally in static intervals, (c) recursive interval merging which locates clusters of varying lengths, and (d) top-down relabeling which concludes the clustering results by re-visiting the output of iterations in (c), as well as their complexities, are described in detail below.

Time Interval Segmentation: The intervals that are used to segment sequences must be chosen appropriately to preserve curve characteristics. We can neither cut one obvious curve feature into two time intervals, nor overly segment them to make features fragmentary. Therefore, we have to extract the main features of the temporal sequences to ensure that the comparison of complicated real-world time series data is reasonable and general. Suppose we have n temporal sequences, $L_1, ..., L_n$, then we first normalize them by subtracting the mean value in the sequences and dividing all values by the standard deviation to extract the trends by enhancing the shape aspect of the sequences [11]. This normalization may enhance local features by exaggerating their curvatures, which is also commonly observed in SIS clustering [13]. We then approximate each normalized sequence with a series of line segments with piecewise linear representation (PLR). In this specific case, we choose Douglas-Peucker algorithm (DP) [9] for its accuracy. This PLR procedure could automatically neglect the unwanted noise, and obtain major developments in sequences.

Thereafter, we need to partition the entire period based on the approximated sequences, denoted by $l_1, ..., l_n$, into m time intervals for further usage. We denote $s_{i,j}$ as the jth unit segment of the ith sequence where i ranges from 1 to n, and j from 1 to m. Since time points have practical meanings, we choose the unit segment splitting points identically for all the sequences to ensure time-series correspondence. Specifically, we gather all the inflexion points in the approximate sequences (i.e., $l_1, ..., l_n$) into one set, and analyze their distributions along the time axis. We first delineate several ranges in which a large number of inflexion points fall. Then, in every range, we take the average position of the in-range points as the position for a splitting point. In this way, the points selected could reasonably reveal the most common transition points for the overall sequence trends, and thereby split the original sequences into distinct time intervals that best preserve sequence characteristics. These time intervals form a set S_i, and are then used as the *basic time intervals*, which are used to generate unit segments (i.e., $s_{i,j}$) for all sequences.

Per-interval Clustering: To find natural clusters in each basic interval (C.1), we group all n layers in each time interval j independently into k clusters using DBSCAN [17], with k varying in each time segment. We choose DBSCAN because it is the most representative density-based clustering algorithm, and it could cluster data into an appropriate number of groups automatically as long as the variables for fault tolerance are specified. Most clustering methods (e.g., K-means) require specifying the cluster number, which is unrealistic to be determined subjectively for all basic intervals without foreseeing data complexity in each time segment. As for the distance measurement in DBSCAN, we use dynamic time warping (DTW) [23], which is an algorithm for measuring similarity between two temporal sequences. To make sure our algorithm reveal relatively significant result, the *MinPts* for DBSCAN (i.e., the minimum data objects required inside the cluster) is set to be one tenth of the overall number of total segments in an interval. With *MinPts* set, we follow the conventional approach introduced in [25] for choosing *Eps* (the radius of the cluster): for some *MinPts*, we find the *MinPts*-th nearest point's distance for every point, and sort them in increasing order. There is generally a point with *MinPts*-th nearest point's distance being distinctly large, and the last *MinPts*-th distance which is not drastically different becomes *Eps*.

Recursive Interval Merging: In the previous step, all unit segments of layers are clustered inside individual basic intervals independently. Therefore, cluster durations are also bound by basic intervals. However, in real applications, one cluster may easily cover multiple consecutive basic intervals. Thus, we would like to recover the cluster duration information (C.2) by recursively merging basic intervals. By merging, we intend to find layer clusters that have a long duration by comparing clustering results in consecutive basic intervals.

The basic idea is to enumerate all possible ways of grouping the basic intervals, and find the best grouping strategy based on a cost function. Practically, the calculation is recursive and may be represented as a tree (Fig. 3). Each node in the tree represents a way to partition the sequence of basic intervals. A parent-child edge represents adding one more partition point to the partition strategy in the parent node. The tree keeps growing until none of the tree nodes can be further partitioned. Therefore, a grouping strategy is equivalent to a leaf node. We use dynamic programming to find the best strategy. For convenience, we denote $l_{p,i}^{a,b} = \{s_{i,a}, ..., s_{i,b}\}$ as the segment of the ith layer in the pth partition level (i.e., being divided into p segments) formed by unit segments $s_{i,a}, ..., s_{i,b}$, and $G_{p,i}^{a,b}$ as the clustering result of these segments. $G_{p,1}^{a,b}, ..., G_{p,n}^{a,b}$ form $G_p^{a,b}$, which is the clustering results for all the segments $s_{i,j}$ in the pth partition level.

Starting from every segment at partition level one (i.e., $l_{1,i}^{1,m}$), we recursively divide the segment into two sub-segments: $l_{p+1,i}^{a,s} = \{s_{i,a}, ..., s_{i,s}\}$ and $l_{p+1,i}^{s+1,b} = \{s_{i,s+1}, ..., s_{i,b}\}$, where $a+1 \leq s \leq b-1$, until $l_{i,p}^{a,b}$ could be partitioned into two unit segments in basic time intervals or if it is a unit segment itself (i.e., $b = a+1$ or $b = a$). We compare them by computing the intersection between every two clusters with a fault tolerance. Specifically, for segment $l_{p,i}^{a,b}$,

if its two sub-segments $l^{a,s}_{p+1,i}$ and $l^{s+1,b}_{p+1,i}$ are in the same cluster as the corresponding sub-segments from another segment $l^{a,b}_{p,j}$, or if it is similar to the representative curve of a cluster, it is allocated to a cluster; otherwise, it is marked as *unclassified*. Allocations for all the involved segments constitute $G^{a,b}_p$. $N^{a,b}_p$ denotes the number of clusters that $G^{a,b}_p$ contains.

Since we would intend find the most general patterns (C.1), we greedily select the partition position s that could (1) minimize the number of un-clustered segments and (2) result in the smallest possible number of clusters under the condition of (1). This condition ensure that the algorithm neither clips the layers into to many segments, nor generates too many clusters at a time point:

$$s = \textbf{argmin}(N_{unclassified}, N^{a,b}_p)$$
$$G^{a,b}_p = \textbf{merge}(G^{a,s}_{p+1}, G^{s+1,b}_{p+1})$$

Top-down Re-labeling: Once the best grouping strategy is calculated, we use the result to combine the obtained clustering results G in each grouping level, and to change the layer segment labels accordingly (C.2). First, all unit segments are labeled *unclustered*. We start traversing from $p = 1$ to the maximum p recorded in the merging step to allocate all the unit segments. Specifically, for a layer l_i, if its unit segment $s_{i,j}$ is marked *unclustered* at level p and the cluster it belongs to at that level still contains enough unclassified segments, then we mark its clustering result as its allocation in $G_{i,p}$.

Complexity Analysis: The complexities for the most steps are intuitive: time interval segmentation and per-interval clustering are bound by the complexity of DP algorithm and DBSCAN respectively, both of which are $O(nlogn)$. The top-down re-labeling step will check at most mn segments in each iteration, and therefore results in $O(m^2n)$. The recursive step, on the other hand, will take up the most time. Typically, for an iteration at pth partition level, its complexity could be expressed as

$$T(p) = 2T(p-1) + f(n),\ 1 \le p \le m$$

where $f(n)$, being $O(nlogn)$, denotes the complexity for computing intersections of clusters for two consecutive intervals. This equation leads to the final complexity of $O(2^m nlogn)$. Since *MinPts* is less than one tenth of the total number, m will be less than ten. Thus, it could afford dealing with complex massive data.

6 Visual Design

We develop a visualization method to present the otherwise too abstract clustering results and reveal the connections between accumulated and individual data. Our design follows four design principles, namely **legibility** [4], [32], **aesthetics** [4], [15], **neatness** [29], and **faithfulness** [29], as discussed below.

6.1 Stacked Graph Constructed with Clusters

Instead of visualizing the data layer by layer on the entire period, we place segment clusters generated in clustering processor independently on the time axis. As shown in Fig. 4(b), we ensure that layer segments within a cluster are stacked consecutively and use independently chosen colors to represent clusters. Segments that do not belong to any clusters are colored grey. The number of clusters is normally much smaller than the layer number. Thus, a cluster-based stacked graph could efficiently improve scalability to a certain extent while preserving a reasonable level of detail for individual sequences, thereby improving **legibility**.

Since clusters may be located in the middle of the entire period, ordering clusters with overlapping periods inappropriately could elevate some regions of one cluster abruptly (Fig. 4(a)). Fortunately, information from the clustering procedure could help with the ordering scheme. Since clustering is processed recursively starting from segments on the entire period, leaf nodes (segments allocated to clusters on a certain level) do not overlap with those in their parental levels. Thus, cases in Fig. 4(a) could be avoided by stacking the clusters following their orders of allocation in the top-down relabeling step, and the noises at the top to fill the gaps. This ordering naturally stacks clusters by their durations, i.e., shorter clusters are always placed on top of longer ones. This could ensure our design to be more **legible** and **aesthetically** pleasing.

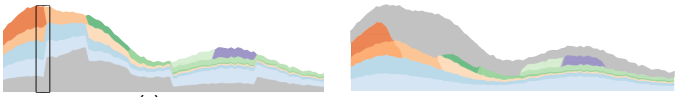

Fig. 4. Comparison of different ordering: (a) due to the sudden changes in the lower cluster, all cluster above encounter a jacked-up effect and become incoherent; (b) properly ordered clusters are smooth, with every color representing a cluster, and grey for outliers.

6.2 Glyphs Encoding Contributions

Although our partitioned stacked graphs could ease clutter problems and indicate trend distributions, the baseline for each cluster is still distorted, and relations between clusters and the aggregation remain unclear. We solve this issue by quantifying the concept of contribution and augmenting the stacked graphs with visual glyphs, and thereby ensure **faithfulness**: we encode the *per-cluster* contribution, which summarizes the contributions from segments in a cluster (i.e., with similar shapes) compactly. We also encode the *average* contribution from all layers.

Specifically, we first compute the *per-layer* contribution. The per-layer contribution for a time point is a comparison between a layer's local shape, which is defined using two consecutive time slices formed by t_i and its neighbors t_{i-1} and t_{i+1}, with the aggregation's. The magnitude and direction of a layer's contribution in a slice (e.g., between t_{i-1} and t_i) are determined separately. The magnitude measurement is inspired by short time series distance [24]. It is the absolute value of the difference between the slopes of the selected layer and the aggregation in the time slice. The direction is set to be positive if the two slopes share the same sign, and negative otherwise. The contribution of the sequence at t_i is the average of the two time slice contributions. We then compute the per-cluster and average ones:

- per-cluster: the mean of per-layer contributions in a cluster,
- average: the mean of contributions of all layers, regardless of cluster information.

Contributions could be encoded in multiple ways, such as glyph directions (upper and lower triangles) or transformations (rotating diamonds). However, for distinguishing the per-cluster and average contribution while still reserving the encoding consistency for the direction and significance of every contribution, we choose to use glyph colors and types (Fig. **??**). We use red and green representing positive and negative contributions respectively, and the saturation for the significance of the contribution. The red-and-green encoding follows the color scheme for stock markets. Grey glyphs reflect that the involved segments, on average, are developed similarly to the aggregation.

We represent per-cluster and average contribution at each time point with diamonds and circles, respectively. Per-cluster

contribution diamonds are located at the middle of each cluster at every time point. The average contribution circles are on the borderline (i.e., aggregated shape of all the layers) of the stacked graph. This borderline is denoted with a dashed line. While in the overview, circles are all placed precisely on the dashed line, they would offset slightly in response to the effects of decomposition. The objectives of the dashed line and the offsets are further described with the interactions (Section 6.3). For visual clearness (i.e., **neatness**), instead of covering the whole stacked graph with diamonds, we only display those with apparent red or green on default (i.e., significant contributions), while the others are shown only by interaction. We later noticed in preliminary reflections that some users still found the visual form too informative even with this pre-defined significance ratio: they argued that the overlaid glyphs could interfere with other patterns. Therefore, for visual simplicity, we decide to offer the option to deliver the glyphs on demand. For instance, users could choose to only display a chosen range of glyphs when the they want to check the contributions for some clusters in some intervals.

6.3 Interactions

Displaying all the information at once is impractical, so we start with only one stacked graph as the overview, and then we provide the following three interactions for progressive exploration.

Brushing. Brushing on a visual element provides users with basic information, so that they can decide whether to further evaluate the element. Our system provides three brushing targets: layer, cluster, and aggregation. Since all clusters are ordered independently in each time interval, the distribution of every layer across clusters is not apparent. To compensate for the destruction of layer continuity, when users brush a layer segment in any cluster, our system highlights the other segments that belong to this layer over the entire period. Brushing a cluster is equivalent to brushing all segments inside, which could show if this cluster of layers remains a cluster at the other time points. For the aggregation level, all the diamond glyphs along a time point are displayed when the aggregation at that point is brushed, which enables comprehensive comparisons between all the sequences and the aggregation at the point.

Selection. Selection is designed to pinpoint aggregations at certain time points and compare their causes of construction. The similarity of the causes is measured with normal mutual information [6], a typical method for comparing clustering results. For simple yet effective visualization, we draw connections between the time points being compared and color the links to clearly represent similarities (a real number in $[0,1]$), with green being highly different and red being highly similar. When a time point is selected, all of its correlation links are shown to locate regions that share significantly similar or different clustering results.

Decomposition. Users could drag individual sequences or clusters back and forth to construct multiple unique stacked graphs with any combination that they might find interesting. For instance, users could locate a cluster responsible for an aggregated feature, and drag it out to construct another graph to see whether this group also contributes to other interesting patterns. A dashed line is drawn constantly in all the graphs, so to preserve the aggregation shape of all the sequences. The glyphs are drawn independently in each stacked graph, and the circles representing average contributions are offset slightly from the dashed line, with the offset distances encoding the differences with the average contribution of all sequences (Fig. 5). Circles placed higher indicate that data in the graph are more responsible for the aggregation than average (either positively or negatively), and less responsible otherwise.

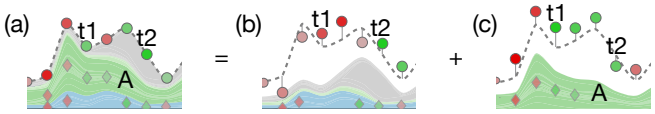

Fig. 5. Example of decomposition. The average contributions at time point t1 and t2 in the complete graph (a) are both negative. After decomposing cluster A, (b) shows the remaining layers have a positive average contribution at t1, and its absolute value is bigger than that in (a). Similarly, (c) indicates that A has a stronger negative contribution to the aggregation at t1. Layers in (b) and (c) hold a stronger and a weaker negative contribution at t2, respectively.

7 Case Study

We apply PieceStack to two datasets in this section. The unemployment rate of industries is mainly to demonstrate basic functionalities, while the Twitter data is more comprehensive for evaluating the usefulness of PieceStack.

7.1 Unemployed US Workers by Industry

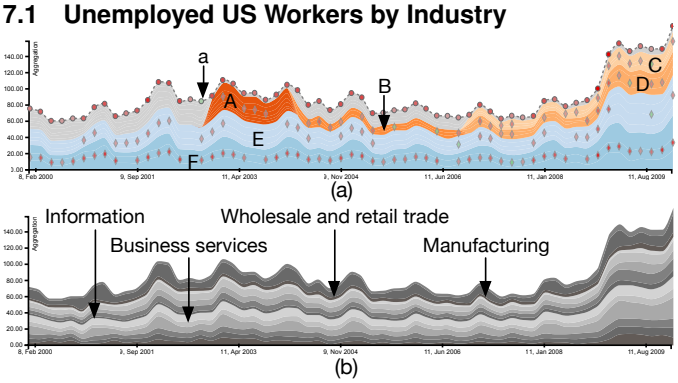

Fig. 6. Overviews of unemployment rates of 14 industries from 2000 to 2010 with (a) PieceStack, with six extracted clusters marked as A-F; (b) traditional stacked graphs for comparison.

We use the unemployment rates of 14 industries from 2000 to 2010 collected by the US Bureau of Labor Statistics to analyze if the rates among industries share similar variations, or if the summation is similar to the individual rate sequences. The industries include mining and extraction (ME), transportation and utilities (TU), wholesale and retail trade (WRT), construction (CON), leisure and hospitality (LH), self-employed (SE), manufacturing (MANU), information (INFO), business services (BS), government (GOV), education and health (EH), agriculture (AGR), finance (FINA) and others (O).

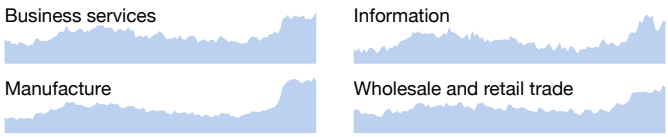

Fig. 7. Expanded small multiples for sequences in cluster E. The similar wave-like trend shows that the clustering is reasonable.

In the PieceStack overview (Fig. 6(a)), we notice that there are six clusters extracted. Two of them are over the entire period (clusters E and F) representing stable correlations of unemployment rates. The other clusters are in partial time intervals (clusters A-D), indicating similarities only during a short period. To demonstrate the clustering result, we expand cluster E and draw the contained

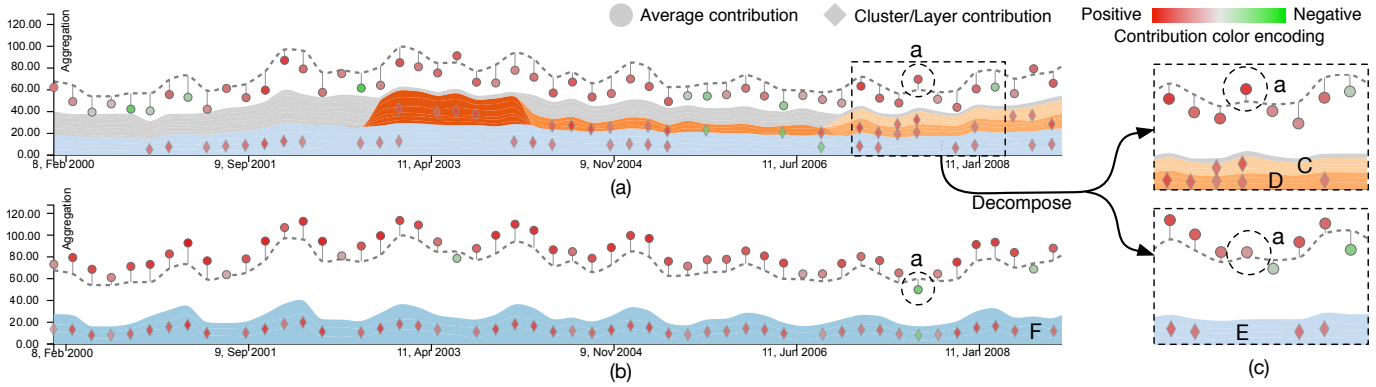

Fig. 8. Decomposition results: (a) the graph with F filtered out; (b) the graph only containing cluster F; (c) the decomposition of (a).

industries in Fig. 7. We confirm they do share similar trends. However, this is unapparent even if we clearly indicate these industries on the traditional stacked graph (Fig. 6(b)).

The glyphs representing contributions are also informative. Observing the distribution of diamonds, we notice that cluster F overlays red diamonds almost everywhere, whereas other clusters display a significantly smaller number of less saturated diamonds. This means that sequences in F are the primary contributors to the aggregation. We are thus interested in the effect on the aggregation if these industries in F are filtered out. We decompose the stacked graph into two sub-graphs, one only containing cluster F (Fig. 8(b)) and the other consisting of all the remaining clusters (Fig. 8(a)). Comparing the offsets of the circles, we find that filtering out these industries can significantly reduce the salience of the local bumps, and the remaining industries have a much lower contribution overall. However, some exceptions occur. For instance, the average contribution at point a increases in Fig. 8(a) and decreases to negative in Fig. 8(b). This suggests that the small bump is not caused by cluster F. Further decompose cluster E, we can clearly see from the results in Fig. 8 that this bump is formed by three groups, F, E, and C+D. Specifically, cluster F has a green circle that indicates negative contribution or inverted trend to the aggregated bump. Cluster E has a light red circle that indicates its trend is subtle but aligned with the average local trend. Cluster C and D are the main causes of the bump, since the figure shows a dark red circle with a positive offset. Though the bump itself is not significant enough to raise public concerns, it may worth further investigation since it is an outlier pattern. However, this pattern is extremely hard to find in Fig. 6(b).

7.2 Twitter Data about Ebola

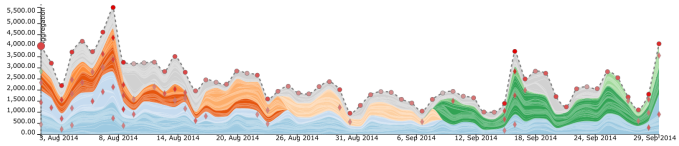

Fig. 9. Overview of the top 1000 most popular hashtags related to Ebola in August and September, 2014. Eight clusters are identified with different colors.

A total of 1,175,263 tweets containing the word "Ebola" are extracted for Aug and Sep 2014, during which the Ebola pandemic reached a climax. We analyze the hashtag distributions to learn the degree of public interest on Ebola and their trigger events. Fig. 9 shows the overview of the top 1000 frequently occurring hashtags with "#ebola" filtered out. However, initial explorations show that most hashtags are neither significant enough to affect the aggregated shape, nor do they attract attentions for further analysis. Nevertheless, we observe that the top 150 hashtags (Fig. 10) are significant enough for the overall trend. Thus, for narrative clarity, we use these hashtags in the following discussions.

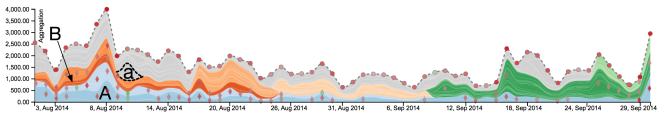

Fig. 10. Overview of the top 150 popular hashtags. The overall trend is similar to the one in Fig. 9.

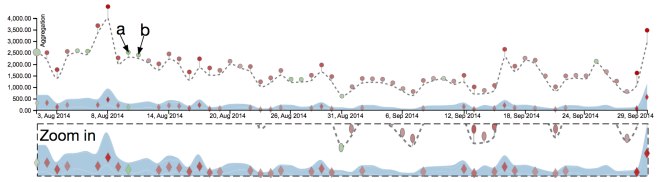

Fig. 11. Cluster A from Fig. 10, the only cluster defined on the whole time period. The red circles indicate that this cluster has positive contributions to the overall shape at most time points.

Fig. 10 shows there is one cluster (A in Fig. 10) generated with respect to the entire period. Extracting this cluster (Fig. 11), we observe that its shape is mostly aligned with the overall aggregation (i.e., the dashed line). A further investigation of the cluster shows that its main members include "#ebolaoutbreak" and "#breaking", which are the two most commonly tagged hashtags (other than "#ebola") used to denote that a tweet is related to Ebola (e.g., "President Ellen Sirleaf : we need HOPE not fear. #ebolaoutbreak"), and therefore could represent the overall popularity of Ebola on Twitter. However, exceptions still exist. For instance, in Fig. 11, the average contributions at location a and b are negative. In contrast to the overall mild bump at these two locations, the blue cluster shows a dent. We go back to the overview and, after a few interactions, we discover that the bump is caused by the highlighted outlier hashtag "#forwardnigeria" (a, Fig. 10). Twitter users decided to use this hashtag on Aug. 10 to promote important facts about Ebola, such to help clarify the almost concurrently widespread rumors in Nigeria about curing Ebola with salt water. Brushing this layer shows it is a brief focus and almost remains untagged at all other time points.

To compare with this entire-period cluster A, we extract the orange cluster (B, Fig. 10). As in Fig. 12, the layers are not clustered during Aug. 26 and Sep. 8, but are split into cluster C and D later. We also find that this cluster generally has positive contributions during period T_1 and T_3 (the negative contributions in T_4 are caused by the outlier hashtag "#forwardnigeria" as discussed before) and irregular contributions in period T_2. This may be attributed to the relatively more unexpected events encountered in

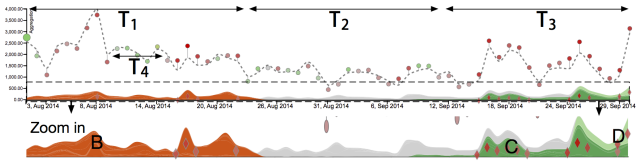

Fig. 12. A stacked graph formed by layers contained in cluster B. This graph generally has positive contributions in T_1 and T_3 and irregular contributions in T_2.

period T_1 and T_3 than T_2. For instance, "#zmapp", the experimental medicine for Ebola, was first discussed in early August when the US government used it on two patients (e.g., "US has #ebola cure. Just letting Africans die! #zmapp") and when the World Health Organization (WHO) endorsed the use of ZMapp on Aug. 12 to combat Ebola. It was re-visited when Liberia announced to receive ZMapp from the US on Sep. 13, and when the first case was discovered in America on Sep. 30 (" #Ebola disease confirmed in @Dallas. I wonder if he will get #zmapp").

"#forwardnigeria" is not the only bursting hashtag triggered by short-term events. Filtering out alike ones, the remaining hashtags that stably occur overall (i.e., aspects that were constantly concerned) are obtained, as in Fig. 13. We can see the aggregated shape is affected greatly: bumps at time points when bursting hashtags appear (d, e, f, and g) disappear. However, main peaks a, b, and c remain. All clusters at these three points have positive contributions (i.e., red circles). However, the green correlation links (h and i) between them indicate that their clustering results are different. For instance, for bumps on Aug. 8 and Sep. 16 (a and b, Fig. 13), when referring back to the texts, we find that their dissimilarity is because their trigger events are different. WHO declared the Ebola epidemic an international health emergency on Aug. 8, which attracted considerable attentions and increased the demands for information regarding the prevention and treatment of Ebola. Thus, related hashtags (e.g., "#ebolavirus", "#howto-preventebola", and "#factsonebola") raised to form a bump. In contrast, bump b was due to President Obama's announcement that the US military would be sent to West Africa to combat Ebola. Twitter users were concerned about this decision's effect on the conflict between America and ISIS with hashtags such as "#usa," "#military," "#isis" ("No troops on the ground to fight #isis, instead let's send our heroes to fight Ebola?").

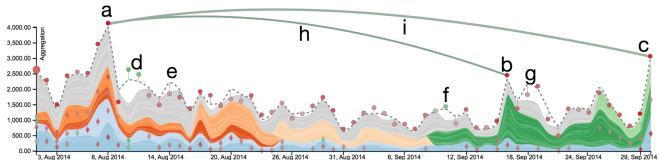

Fig. 13. Hashtag visualization with one-time outliers filtered out.

8 User Study

To further demonstrate the effectiveness of PieceStack, we conducted a comparative study with a benchmark system formed by a classical stacked graph. In the benchmark system, users can drag layers freely, place layers with respect to their interests, or arrange layers around their preferred baselines directly.

Twenty experienced computer users were recruited, including 13 males and 7 females. Their average age was 23.5 (from 21 to 29, median 23). 6 of them reported to had seen a stacked graph before, and one was familiar with time series visualization techniques. A compact dataset containing twelve layers (60 time points for each layer) with clear shape patterns was chosen for this user study. We use the same dataset in both systems to ensure the task difficulties are not affected by patterns' significances. For each user session, we started with a brief tutorial of both systems, and encouraged the users to freely try and to ask questions. They were then asked to finish five simple yet representative tasks with both systems. The tasks were designed for verifying our system's capability of solving the problems summarized in Section 3:

- T.1 Find the most contributive layer(s) for an aggregated bump;
- T.2 Find the types of contributions (significantly positive, significantly negative, or trivial) of all layers to an overall bump;
- T.3 For two aggregated bumps, find the layers that have the same type of contributions;
- T.4 Find the layer(s) that contributes the most to the overall aggregated shape on the whole time period;
- T.5 For a given layer, find layers that are similar to it at least in some time intervals.

We originally designed it to be a cross-over study to eliminate potential bias. However, our initial experiments showed that users starting with PieceStack significantly out-performed those following the inverse order when using the benchmark system, while their performances with PieceStack stayed identical. These users reflected that since the layers were specifically organized in PieceStack, they were more informed on some tasks when moving to the benchmark one. For instance, on T.2, with the knowledge from clusters and glyphs, they would consciously and confidently look for a certain number of layers in the benchmark to fit into different types of contributions quickly. In contrast, the layer placements in benchmark hid this information. Given this situation, we decided to follow the benchmark system to PieceStack order, such that the ordering would not introduce additional bias.

We analyzed the answers and the time that the participants spent on each task with both systems. Overall, our system guaranteed improvements in accuracy and efficiency. While the average time for completing each task are shortened approximately by half using PieceStack (with T.4 being an exception, improving from 109.384s to 30.458s), the performance improvement varied more: for easy tasks like T.1, most users provided correct answers with both systems, and the accuracy is only increased from 99.58% to 100.00%, while it raised from 85.83% to 99.17% for the more complicated T.3. Fig. 14 shows more concrete comparisons.

Several subjective measures for PieceStack were also presented, with respect to the overall effectiveness, and the usefulness of some PieceStack designs involved in completing the tasks. All the measures were rated on a five-point scale, with 5 being strongly positive, and 1 strongly negative. Users' responses showed that they found Benchmark slightly more understandable (rated 4.35 versus 4.25 on average, with SD both being 0.75), and PieceStack easier to use (mean $= 4.1$, SD $= 0.79$ versus mean $= 4$, SD $= 1.17$). The significant rate difference for the usefulness problem conveyed that they agreed PieceStack to be much more useful (mean $= 4.7$, SD $= 0.47$ versus mean $= 2.9$, SD $= 0.72$). In addition, all encodings in PieceStack were positively rated, with clustering (mean $= 4.7$, SD $= 0.47$) and decomposition (mean$= 4.7$, SD $= 0.57$) being the most preferred ones and brushing (mean $= 4.25$, SD $= 1.02$) the least. Glyph was rated 4.5 on average with SD $= 0.61$.

To further understand what caused these statistical and subjective differences, we analyzed the approaches users chose to

solve the tasks. For the benchmark system, users' exploration processes were mostly identical. They would repeatedly reorder and stack the others on the layer of interest to compare it with the aggregation. Meanwhile, the usage patterns of PieceStack varied more due to the extra cognitive load caused by the richer visual encodings. For example, only 12 users managed to take advantage of the glyphs (supposedly providing enough information) in T.1 and T.2, and 17 in T.3 and T.4. Others either chose to directly observe the cluster height at each time point (one for T.1 and T.2), or decomposed a whole cluster to compare its shape with the dashed line. In their self-reflections, three users felt that though admittedly the stacked graph was not expressive enough for the tasks, the information in PieceStack was, by contrast, overwhelming. They would need more time to get familiar with the complicated encodings and sometimes would still go with direct observation or decomposition. Only when encountering more difficult tasks T.3 and T.4, they would try to consider the best approach possible and to turn to the glyphs. For the rest who did not use glyphs in the whole process, they reported that they were attracted by the clustering results, which alleviated the baseline distortion and reduced the number of layers they must consider, and therefore ignored the glyphs. Despite the complexity, all the users found some approaches, though not necessarily the most ideal ones, that could make the comparisons among layers and aggregations more intuitive and less time consuming with PieceStack.

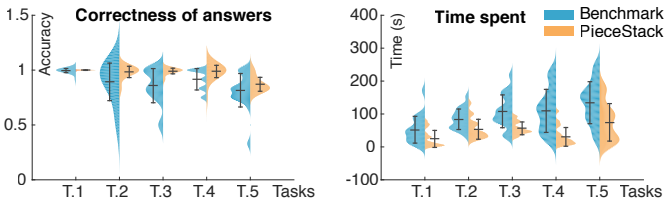

Fig. 14. Violin plot and error bars for the comparisons between the accuracy and efficiency of users completing tasks with two systems. The p-value for their accuracy differences are 0.165, 0.010, $1.621 \cdot 10^{-4}$, 0.001, 0.066 for T.1 to T.5 respectively, and the ones for the time spent are $3.135 \cdot 10^{-4}$, $9.135 \cdot 10^{-6}$, $3.003 \cdot 10^{-6}$, $1.203 \cdot 10^{-5}$, $1.568 \cdot 10^{-5}$.

Overall, the users preferred our system since it provided an overview that could brief them with the general characteristics of stacked graphs and implemented interactions that allowed them to target some parts with potential patterns. For instance, they agreed that the glyphs and cluster colors quantified the comparisons clearly. By contrast, the classical system only left them to explore blindly. The users further made some valuable points regarding of the clustering results' correctness and the intuitiveness of PieceStack. Four users felt when the number of layers reached the scale of hundreds or thousands, stacking them together would compress the layer so greatly that no information left except the aggregated values. In that sense, grouping layers into a number of clusters reasonable enough for display would require a coarser granularity process, and the similarity of layers in the same cluster could be questionable. Therefore, besides the general guideline from the initial clustering result, they recommended to let the users decide the time intervals for more precise and reliable clustering.

9 Discussion

PieceStack is designed to uncover the rich implications of temporal data in stacked graphs. To help users with some common but important tasks, the basic visual settings of stacked graphs are altered and several new elements are added to improve the expressiveness of traditional stacked graphs. The case studies and user study confirm that our system can effectively reveal deep relations among layers and the overall shape.

Compared with other time series visualization such as line charts or small multiples, ours enhances the comparison between layers and aggregations. First, neither of the aforementioned methods encode the aggregated value. Even if the summation is further appended, they are still not suitable for finding the causalities between layers and the aggregation: the overlaid line chart with large line collections suffers greatly from the occlusion issue, and the comparisons between the aggregation and individual lines (i.e., layers in stacked graphs) are not intuitive. As for small multiples, since every layer is drawn in an independent view, understanding the cause of an aggregation will require the users to go through every view and understand each layer's contribution individually, which would be cumbersome.

Although useful and effective, PieceStack has some design limitations. First, even if PieceStack is theoretically capable of processing all types of time series data, it prefers those with more apparent similarities and differences. If most layers evolve uniquely and randomly, the clustering algorithm is likely to declare many noises. In this situation, our system may not work better than traditional stacked graphs, where the layers are colored distinguishably. This can be alleviated by adjusting some detailed steps, such as normalization and distance calculation, to cater to the features of a specific dataset. Besides, the decomposition visualization may become less clear when the extracted layers are not comparable to the overall aggregation. In our design, the dashed lines and extracted layers are located in the same scale to aid comparison. However, if the layers are too small, they will be perceptually too flat for users to recognize features. To solve this, non-linear scaling or fisheye interactions can be adopted to help users observe detailed layer shapes.

10 Conclusion and Future Work

In this paper, we propose PieceStack, an interactive visual system designed to help users understand the formation of stacked graphs. First, we derive a set of design requirements to address some empirically categorized questions concerning the relations among layers and the aggregation. Guided by these requirements, a new algorithm is proposed to discover partial similarities between temporal sequences and to support our visual design. Based on stacked graphs, our visualization provides a comprehensive method to help users analyze and evaluate how layers contribute to the overall shape. Case studies and a user study exemplify the convenience and effectiveness of PieceStack for understanding individual and aggregated patterns of temporal sequences. In the future, we aim to adjust the clustering algorithm with time series data from various fields, and to increase its capability to reveal data patterns in different forms. We will also further implement zoom operations to help evaluate thin layers and improve the scalability. Furthermore, we plan to customize the exploration intervals and enable users to choose the time interval they would like to further partition, cluster, and visualize to glean detailed local patterns.

Acknowledgments

We gratefully thank all the participants for their feedbacks during the user study, and the anonymous reviewers for their valuable and constructive comments.

REFERENCES

[1] C. Abraham, P.-A. Cornillon, E. Matzner-Løber, and N. Molinari. Unsupervised curve clustering using b-splines. *Scandinavian journal of statistics*, 30(3):581–595, 2003.

[2] D. Baur, B. Lee, and S. Carpendale. Touchwave: kinetic multi-touch manipulation for hierarchical stacked graphs. In O. Shaer, C. Shen, M. R. Morris, and M. S. Horn, editors, *ITS*, pages 255–264. ACM, 2012.

[3] S. Bruckner and T. Möller. Result-driven exploration of simulation parameter spaces for visual effects design. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):1468–1476, 2010.

[4] L. Byron and M. Wattenberg. Stacked graphs–geometry & aesthetics. *Visualization and Computer Graphics, IEEE Transactions on*, 14(6):1245–1252, 2008.

[5] N. Cao, L. Lu, Y.-R. Lin, F. Wang, and Z. Wen. Socialhelix: visual analysis of sentiment divergence in social media. *Journal of Visualization*, 18(2):221–235, 2015.

[6] K. W. Church and P. Hanks. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29, 1990.

[7] W. Cui, S. Liu, L. Tan, C. Shi, Y. Song, Z. Gao, H. Qu, and X. Tong. Textflow: Towards better understanding of evolving topics in text. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2412–2421, 2011.

[8] M. Dork, D. Gruen, C. Williamson, and S. Carpendale. A visual backchannel for large-scale events. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):1129–1138, 2010.

[9] D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.

[10] S. Fröhwirth-Schnatter and S. Kaufmann. Model-based clustering of multiple time series. *Journal of Business & Economic Statistics*, 26(1):78–89, 2008.

[11] S. J. Gaffney. *Probabilistic curve-aligned clustering and prediction with regression mixture models*. PhD thesis, University of California, Irvine, 2004.

[12] A. Gisbrecht. Time series clustering. *ICOLE 2007, Lessach, Austria*, page 48.

[13] D. Goldin, R. Mardales, and G. Nagy. In search of meaning for time series subsequence clustering: matching algorithms based on a new distance measure. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 347–356. ACM, 2006.

[14] S. Hangal, M. S. Lam, and J. Heer. Muse: Reviving memories using email archives. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 75–84. ACM, 2011.

[15] S. Havre, E. Hetzler, P. Whitney, and L. Nowell. Themeriver: Visualizing thematic changes in large document collections. *Visualization and Computer Graphics, IEEE Transactions on*, 8(1):9–20, 2002.

[16] J. Heer, F. B. Viégas, and M. Wattenberg. Voyagers and voyeurs: Supporting asynchronous collaborative visualization. *Communications of the ACM*, 52(1):87–97, 2009.

[17] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[18] E. Keogh and J. Lin. Clustering of time-series subsequences is meaningless: implications for previous and future research. *Knowledge and information systems*, 8(2):154–177, 2005.

[19] J.-G. Lee, J. Han, and K.-Y. Whang. Trajectory clustering: a partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 593–604. ACM, 2007.

[20] T. W. Liao. Clustering of time series data: a survey. *Pattern recognition*, 38(11):1857–1874, 2005.

[21] S. Liu, M. X. Zhou, S. Pan, Y. Song, W. Qian, W. Cai, and X. Lian. Tiara: Interactive, topic-based visual text summarization and analysis. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(2):25, 2012.

[22] B. Morris and M. Trivedi. Learning trajectory patterns by clustering: Experimental studies and comparative evaluation. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 312–319. IEEE, 2009.

[23] M. Müller. Dynamic time warping. *Information retrieval for music and motion*, pages 69–84, 2007.

[24] C. S. Mller-Levet, F. Klawonn, K.-H. Cho, and O. Wolkenhauer. Fuzzy clustering of short time-series and unevenly distributed sampling points. In M. R. Berthold, H.-J. Lenz, E. Bradley, R. Kruse, and C. Borgelt, editors, *IDA*, volume 2810 of *Lecture Notes in Computer Science*, pages 330–340. Springer, 2003.

[25] T. Pang-Ning, M. Steinbach, V. Kumar, et al. Introduction to data mining. In *Library of Congress*, page 74, 2006.

[26] W. Playfair. Commercial and political atlas and statistical breviary, 1786.

[27] C. A. Ratanamahatana, J. Lin, D. Gunopulos, E. J. Keogh, M. Vlachos, and G. Das. Mining time series data. In *Data Mining and Knowledge Discovery Handbook, 2nd ed.*, pages 1049–1077. 2010.

[28] C. Shi, W. Cui, S. Liu, P. Xu, W. Chen, and H. Qu. Rankexplorer: Visualization of ranking changes in large time series data. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12):2669–2678, 2012.

[29] C. Shi, Y. Wu, S. Liu, H. Zhou, and H. Qu. LoyalTracker: Visualizing loyalty dynamics in search engines. *IEEE Transactions on Visualization and Computer Graphics (Proceedings of IEEE VAST 2014*, 20(12), 2014.

[30] A. Singhal and D. E. Seborg. Clustering multivariate time-series data. *Journal of chemometrics*, 19(8):427–438, 2005.

[31] G. Sun, Y. Wu, S. Liu, T.-Q. Peng, J. J. Zhu, and R. Liang. Evoriver: Visual analysis of topic coopetition on social media. *Visualization and Computer Graphics, IEEE Transactions on*, 20(12):1753–1762, 2014.

[32] Y. Tanahashi and K.-L. Ma. Design considerations for optimizing storyline visualizations. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12):2679–2688, 2012.

[33] F. B. Viegas, M. Wattenberg, F. Van Ham, J. Kriss, and M. McKeon. Manyeyes: a site for visualization at internet scale. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1121–1128, 2007.

[34] X. Wang, K. Smith, and R. Hyndman. Characteristic-based clustering for time series data. *Data Mining and Knowledge Discovery*, 13(3):335–364, 2006.

[35] Y. Xiong and D.-Y. Yeung. Time series clustering with arma mixtures. *Pattern Recognition*, 37(8):1675–1689, 2004.

[36] H. Zhou, P. Xu, and H. Qu. Visualization of bipartite relations between graphs and sets. *Journal of Visualization*, 18(2):159–172, 2015.

Tongshuang WU is currently an undergraduate student in the Department of Computer Science and Engineering at the Hong Kong University of Science and Technology (HKUST). Her research interests are in information visualization, visual analytics, and human-computer interaction, and online education. For more information, please visit http://tongshuang.me/

Yingcai WU is an assistant professor at the State Key Lab of CAD & CG, Zhejiang University, Hangzhou, China. He received his Ph.D. degree in Computer Science from the Hong Kong University of Science and Technology (HKUST). Prior to his current position, Yingcai Wu was a researcher at the Internet Graphics Group in Microsoft Research Asia, Beijing, China. His primary research interests lie in visual behavior analytics, visual analytics of social media, visual text analytics, uncertainty-aware visual analytics, and information visualization. For more information, please visit http://www.ycwu.org

Conglei SHI is a postdoctoral research fellow in IBM T.J. Watson Research Center. He received his Ph.D. degree in Computer Science from the Hong Kong University of Science and Technology (HKUST) and his B.Sc. degree in Shanghai Jiao Tong University in major of Computer Science. His main research interests are information visualization, visual analytics , and human computer interaction. For more information, please visit http://www.conglei.org/

Huamin QU is a professor in the Department of Computer Science and Engineering at the Hong Kong University of Science and Technology. His main research interests are in visualization and computer graphics, with focuses on urban informatics, social network analysis, e-learning, and text visualization. He obtained a BS in Mathematics from Xi'an Jiaotong University, China, an MS and a PhD in Computer Science from the Stony Brook University. For more information, please visit http://www.huamin.org

Weiwei CUI Weiwei Cui is a Lead Researcher at Microsoft Research Asia, China. He received his PhD in Computer Science and Engineering from the Hong Kong University of Science and Technology and his BS in Computer Science and Technology from Tsinghua University, China. His primary research interests lie in visualization, with focuses on text, graph, and social media. For more information, please visit http://research.microsoft.com/en-us/um/people/weiweicu/