# CorrelatedMultiples: Spatially Coherent Small Multiples with Constrained Multidimensional Scaling

Xiaotong Liu, Yifan Hu, Stephen North, Teng-Yok Lee, Han-Wei Shen

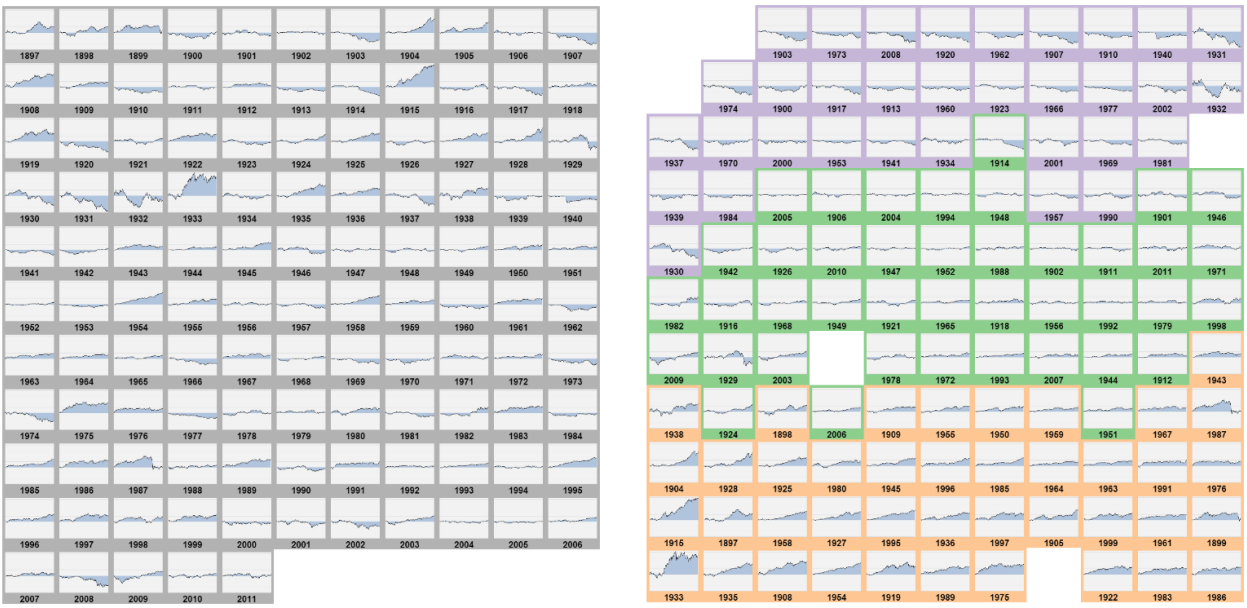

Fig. 1. The Dow Jones Industrial Average (DJIA) from 1897 to 2011. Left: rendered as sequential small multiples, ordered by year. Right: CorrelatedMultiples, in a spatially coherent layout based on similarity. Charts of the years 2008 and 1920 are similar and are placed close to each other at the top of CorrelatedMultiples (right), but are far apart in the sequential small multiples (left).

**Abstract**—Small multiples are a popular method of summarizing and comparing multiple facets of complex data sets. Since they typically do not take into account correlations between items, serial inspection is needed to search and compare items, which can be ineffective. To address this, we introduce CorrelatedMultiples, an alternative of small multiples in which items are placed so that distances reflect dissimilarities. We propose a constrained multidimensional scaling (CMDS) solver that preserves spatial proximity while forcing items to fit within a fixed region. We evaluate the performance of CMDS in comparison with competing methods, and demonstrate the effectiveness of CorrelatedMultiples for visual search and comparison through a controlled user study and two real-world applications.

## 1 Introduction

As data from public, commercial, and private sources become increasingly accessible, and the types of information gathered by social networks such as Twitter and Facebook expand, big data are flooding in at a rate never seen before. Large pools of data are being applied to decision making almost in all business sectors. For example, retailers such as Tesco and Fresh Direct gather and analyze transaction data to make decisions about pricing, promotions, and shelf allocation. Ford Motor, PepsiCo, and Southwest Airlines analyze consumer postings on social-media sites to quickly gauge the impact of marketing campaigns, and to understand consumer sentiment about their brands. Visual analysis of similarities and contrasts among data items can help analysts to monitor, explore and make sense of large amounts of information more easily, hence providing a foundation for better decision-making.

- *Xiaotong Liu, Teng-Yok Lee and Han-Wei Shen are with the Ohio State University, E-mail: {liuxiaot,leeten,hwshen}@cse.ohio-state.edu.*
- *Yifan Hu and Stephen North are with AT&T Labs Research, E-mail: {yifanhu,north}@research.att.com.*

Small multiples, described by Jacques Bertin and popularized by Edward Tufte, allow one to examine multiple facets of complex data sets, and support visual comparisons and tracking of dynamic objects [4, 31]. They have been applied to monitoring and analyzing data-intensive processes such as system management, quality control, medical record analysis, and large-scale industrial and engineering operations. Small multiples alleviate overplotting and occlusion that may occur when objects are too close and overlap. Although lining up multiple visualizations allows a viewer to compare plots more easily, as the number and complexity of the items increases, the effectiveness of small multiples quickly diminishes. This is because the perceptual hindrance that forces the user to examine each individual item sequentially rather than obtain a quick overview of the whole data would make the visualization hardly any better than a simple data table [19]. Figure 1 (left) shows an example of small multiples of the Dow Jones stock market index from 1897 to 2011. It is hard to tell what patterns predominate across the years. To amend this, it is highly desirable to have a small-multiples visualization that can better organize the multiple plots based on the correlation in underlying data.

In this paper, we propose *CorrelatedMultiples*, a spatially coherent visualization based on small multiples. The goal of CorrelatedMultiples is to exploit similarity among a set of items to determine their layout. By incorporating similarity into the spatial layout, viewers can better judge the overall qualitative characteristics of the data set [19]. Following the idea of standard multidimensional scaling (MDS) [23], we model the relationship of items as a similarity graph, and embed the items so that proximity reflects similarity. The similarity between items is measured by domain-specific metrics, and related items are discovered using clustering algorithms. To lay out the correlated items so that they fit within some fixed display space, we propose a Constrained Multi-Dimensional Scaling (CMDS) model. The final layout is further adjusted to improve its overall appearance by consistent horizontal and vertical alignment.

CorrelatedMultiples allow users to identify similarities and differences in the data more effectively, because related items are placed nearby, and unrelated items are pushed farther away from each another. Aggregates and anomalies can be discovered and examined based on the items' spatial locations, since adjacent items are likely to be similar, making abnormal events easier to identify. We conducted a controlled user study that shows the effectiveness of CorrelatedMultiples in comparison with the conventional sequential small multiples for searching similar items, and demonstrated how the CMDS method outperforms competing methods in preserving spatial proximity. We also present applications of CorrelatedMultiples in two domains: displaying stock market trends in financial market analysis, and Madden-Julian oscillation in climate modeling. The main contributions of this work are:

- We introduce CorrelatedMultiples, which encode data similarity using spatial proximity among items in small multiples.
- We propose an optimization algorithm, based on constrained multidimensional scaling, to create CorrelatedMultiples.
- We evaluate the effectiveness of CorrelatedMultiples through a controlled user study and two real-world applications.

The rest of this paper is organized as follows. In the next section we briefly review related work. Section 3 describes the design of CorrelatedMultiples. Section 4 describes the Constrained Multi-Dimensional Scaling optimization algorithm. In Section 5 we evaluate the proposed approach by means of a user study and a quantitative assessment. Section 6 presents two applications. Finally, Section 7 summarizes the results.

## 2 Related work

Small multiples, introduced by Jacques Bertin [4] and Edward Tufte [31], are widely used in both the scientific literature and the mass media. As the name suggests, a set of charts are arranged in a grid to encourage comparison. Bertin further considered the possibility of reordering the items to highlight interesting relationships. Javed *et al.* [21] showed that small multiples are more efficient than shared-space techniques for comparisons across time series with a large visual span. Archambault *et al.* [1] found that small multiples gave significantly faster performance than animation for understanding dynamic graphs. Robertson *et al.* [27] also showed that small multiples are more accurate and effective than animation for trend analysis. Woods *et al.* [34] organized small multiples in a grid using a spatial layout adapted from spatially ordered Treemap by Wood and Dykes [33]. However, according to the evaluation by Eppstein *et al.* [13], this approach can result in a relatively large displacement. The same study shows that grid map [13], through point set matching outperforms gridded spatially ordered Treemap. However, optimal matching is computationally expensive with the best known algorithms taking $O(n^6 log^3 n)$ time. Our algorithm can produce grid layouts comparable in quality to grid map, but in much less time.

A recent study showed that grouping significantly improves visual search task performance [19]. However, grouping similar items while efficiently utilizing the display space is non-trivial. In most space-filling visualization techniques such as Treemap [29] and Bubblemap [3], relative positions of cells does not reflect similarity. Itoh *et al.* [20] proposed a hybrid space-filling and force-directed layout to visualize multiple-category graphs. Unfortunately, some space between groups is wasted. Our proposal attempts to utilize the available space as much as possible, while at the same time incorporating similarity among items to enhance comparison.

For showing similarity among a collection of objects, multidimensional scaling (MDS) [23] and graph based techniques play a prominent role in various fields, and are relevant in many application areas [15, 16]. The most common formulation of MDS is a stress model [22], which can be solved by majorization [16]. Our solver also uses a Delaunay triangulation as scaffolding that is combined with multidimensional scaling, to maintain the global structure of a layout while removing overlaps. This is similar to the PRISM approach [15], except PRISM does not account for display space constraints. Work on constrained graph layout appeared as early as the 1990's [24, 30]. Dwyer and Koren [8] proposed the DIG-COLA algorithm for directed graph drawing. Their model also incorporates a notion of hierarchy energy, which respects the orientation of directed edges. Dwyer *et al.* [9] attempted to solve constrained graph layout problems with a combination of stress majorization and constrained programming techniques. Constrained optimization also provides the foundation for Dunnart, a constraint-based graph drawing tool [11]. Further work by Dwyer [2, 7, 12] considered fast satisfaction of simple constraints such as linear or circular constraints. Of these, the closest to our work is Dwyer's [9]. The main difference is that Dwyer [9] aims at satisfying orthogonal ordering constraints using a sophisticated quadratic programming algorithm. Our main objective is to satisfy shape constraints, and our approach is easier to implement — we coded a prototype, including Delaunay triangulation, in 614 lines of JavaScript.

## 3 Design of CorrelatedMultiples

This section presents the CorrelatedMultiples design. We outline a set of goals and potential challenges, explain how the proposed design addresses these, and give an overview of a prototype system.

### 3.1 Design Methodology

The idea behind CorrelatedMultiples is to retain the key advantages of small multiples: (1) displaying small instances of data with a consistent representation, allowing side-by-side visual comparison of multiple items; (2) avoiding occlusion and clutter, hence suffering less from over-plotting. (3) making effective use of display space compared with non-space-filling visualizations, such as graphs and maps. In addition, CorrelatedMultiples enhance small multiples by (4) exploiting similarity to determine the spatial proximity of the items.

One of the challenges addressed in this study is to allow the plotting of spatially coherent small multiples in a constrained space. Also, in most of the space-filling visualizations (such as Treemap [29]), small multiples do not encode data similarity in the layout – similar objects can be placed far apart (such as the charts for the year 1954 and 1995 in Figure 1 (left)), which makes it harder to compare and track the trend of data.

To tackle this challenge, we first model the relationship between small multiples as a similarity graph. The similarity between two items is measured by some appropriate domain-specific distance function on the underlying data. Then, we embed the items in the display area so that spatial proximity reflects similarity. Correlated items are discovered by clustering algorithms on the similarity graph, or on the embedding. To lay out the correlated items within the available display space, while maintaining spatial proximity, we propose a constrained multidimensional scaling model described in Section 4. Also, the locations of the items are aligned horizontally and vertically to improve the overall appearance.

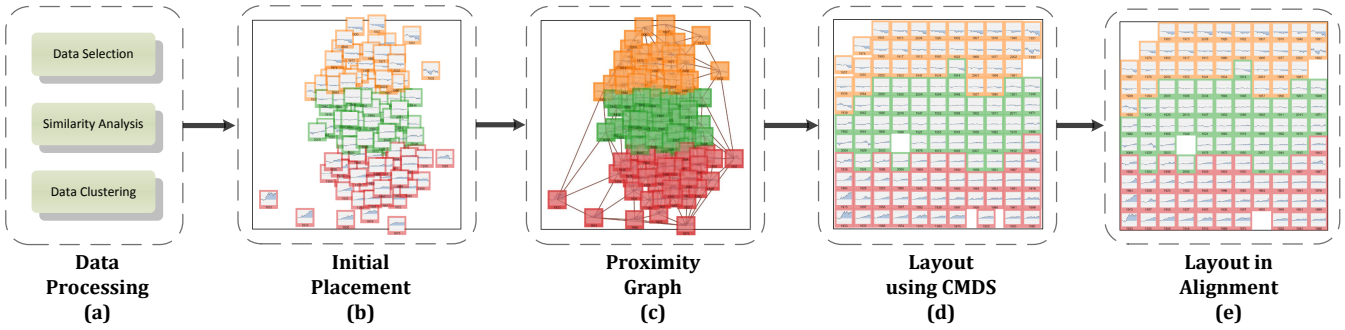


Fig. 2. Overview of the CorrelatedMultiples pipeline: (a) selecting data of interest, analyzing similarity, and finding a similarity-based clustering; (b) placing data items based on similarity (clusters are indicated by color); (c) computing a proximity graph by Delaunay triangulation to constrain relative positions of items; (d) adjusting the layout by the Constrained Multi-Dimensional Scaling (CMDS) algorithm; (e) aligning the final layout horizontally and vertically.

### 3.2 System Overview

Figure 2 shows an overview of our prototype to compute CorrelatedMultiples. Depending on the available display space and the size of each display item, we first select a subset of the most important items that can fit within the display, compute their similarity, and cluster the items (Figure 2(a)). Then the items are assigned initial coordinates by multidimensional scaling (Figure 2(b)). Next, a proximity graph is derived from a Delaunay triangulation of the items (Figure 2(c)). Then a Constrained Multi-Dimensional Scaling (CMDS) model is applied to remove overlaps, while preserving spatial proximity and fitting the visualization to the available display space (Figure 2(d)). Finally, items in the layout are aligned horizontally and vertically to improve the overall appearance (Figure 2(e)).

## 4 Constrained Multidimensional Scaling Algorithm

In this section, we describe the Constrained Multi-Dimensional Scaling (CMDS) algorithm for CorrelatedMultiples. The proposed algorithm is also applicable to general graph visualization.

### 4.1 Model Formulation

In graph drawing, stress minimization, based on multidimensional scaling, has been applied to achieve predefined target lengths in high quality layouts. The full stress model assumes there are springs between all node pairs. Given a 2-D layout where node $i$ is placed at point $p_i$, the energy of this spring system is

$$\sum_{\{i,j\}\in V} w_{ij}(\|p_i - p_j\| - d_{ij})^2, \tag{1}$$

where $d_{ij}$ is the ideal (typically, graph-theoretic shortest path) distance between nodes $i$ and $j$, and $w_{ij}$ is a weighting factor, typically $1/d_{ij}^2$. A layout that minimizes the total stress is considered optimal. We formulate the CMDS model from this basic stress model.

Given a proximity graph $G = (V,E)$, with $V$ the set of nodes and $E$ the set of edges, and a display region $\Gamma$, the goal is to find coordinates $p_i$ for each node $i \in V$, such that: (1) there is no overlap between any nodes $\{i,j\} \in V$; (2) each edge $(i,j) \in E$ is close to its ideal length; (3) each $p_i$ is inside $\Gamma$. The first condition, no-overlap, is needed for readability and aesthetics; the second preserves spatial proximity of nodes in the graph. At the same time, the graph should fit in $\Gamma$ to utilize the available space without being scaled down so much that it reduces readability. In small-multiples, nodes are not just points but have (the same) finite area, so the actual display area $\Gamma$ must be shrunk by the margin of half the node's width and height, so that every node can be fully displayed as long as its center lies within $\Gamma$. With these conditions in mind, we propose the CMDS model

$$\min \sum_{(i,j)\in E} w_{ij}(\|p_i - p_j\| - d_{ij})^2 + \alpha \sum_{p_i \notin \Gamma} (\|p_i - \Gamma(p_i)\|)^2, \tag{2}$$

where $d_{ij}$ is the ideal distance between nodes $i$ and $j$, $w_{ij}$ is a weighting factor, $\alpha \geq 0$ is a parameter to be determined, and $\Gamma(p_i)$ denotes the projection of $p_i$ to the region $\Gamma$ – if $p_i$ is outside of $\Gamma$, the projection $\Gamma(p_i)$ gives the point on the boundary of $\Gamma$ that is closest to $p_i$; otherwise $\Gamma(p_i) = p_i$.

The first term of (2) encodes the stress energy between nodes sharing an edge. The initial value of $p_i$ is determined by a standard MDS embedding, or may be supplied by the user. Subsequently, to maintain the proximity relation, we follow the PRISM approach [15]: we take a Delaunay triangulation of the layout, set the graph $G$ to the proximity graph generated by this triangulation (thus, $E$ is the set of triangulation edges), and solve to preserve node distances along triangulation edges. The rigidity of the triangulation provides sufficient scaffolding to constrain the relative positions of the components, and helps preserve the global structure of the original MDS layout. Fortunately, we only need to consider pairs of nodes that share an edge, which yields a sparse model that can be solved much faster than a full stress energy model (as in (1)). For each edge $(i,j)$, the amount of overlap on the line $x_i \rightarrow x_j$ is denoted by $\delta_{ij}$ ($\delta_{ij}$ is set to 0 if no collision is detected), and the ideal distance is set as

$$d_{ij} = l_{ij} + \delta_{ij}, \tag{3}$$

where $l_{ij}$ is the original distance between node $i$ and $j$.

The second term of (2) is the stress energy between node $i$ and $\Gamma$ if node $i$ is outside of $\Gamma$, and can also be written as

$$\alpha \sum_{p_i \notin \Gamma} (\|p_i - \Gamma(p_i)\| - 0)^2, \tag{4}$$

which sets the ideal distance between node $i$ and its projected point $\Gamma(p_i)$ to 0. By minimizing this combined stress energy, nodes will be constrained within $\Gamma$ as $\alpha$ gets progressively larger, provided that $\Gamma$ is large enough relative to the total node area.

Taking the gradient of (2) with respect to $p_i$, assuming that $\Gamma(p_i)$ is constant, and setting the gradient to zero gives

$$\sum_{j:\ (i,j)\in E} w_{ij}(\|p_i - p_j\| - d_{ij}) \frac{(p_i - p_j)}{\|p_i - p_j\|} + \alpha(p_i - \Gamma(p_i)) = 0. \tag{5}$$

By algebraic manipulation of (5), keeping linear terms involving $p_i$ to the left and moving the rest to the right of the equation, we obtain the following iterative scheme

$$p_i \leftarrow \frac{\sum_{(i,j)\in E} w_{ij}\left(p_j + d_{ij}\frac{p_i - p_j}{\|p_i - p_j\|}\right) + \alpha\Gamma(p_i)}{\sum_{(i,j)\in E} w_{ij} \ + \ \alpha}. \tag{6}$$

While we could apply a stress majorization method [16] to solve (2), we chose the above iterative process because it does not require the

solution of a linear system. This makes it easier to implement in languages like JavaScript that do not have sophisticated numerical libraries. More importantly, by rendering the iterative process, we can make a visually stable animation, from an initial unconstrained configuration, to the final constrained layout. This is shown in the video that accompanies this paper.

After $N$ iterations of (6), the layout may still have overlaps and nodes outside display space. If so, we regenerate the proximity graph using a Delaunay triangulation augmented with additional edges for overlapping nodes, calculate ideal edge lengths, and rerun the energy minimization step. When the total stress no longer decreases, the final layout is aligned horizontally and vertically using a rounding function:

$$\begin{cases} x_a = round((x - w/2)/w) \times w + w/2 \\ \\ y_a = round((y - h/2)/h) \times h + h/2 \end{cases}, \tag{7}$$

where $x$ and $y$ are the coordinates of the center of a node, $w$ and $h$ are the width and height of the node, $x_a$ and $y_a$ are the aligned center of the node, and the *round* function gives the integer that is closest to the input value. We refer to this as the Constrained Multi-Dimensional Scaling (CMDS) algorithm, shown in Algorithm 1.

**Algorithm 1** Constrained Multi-Dimensional Scaling (CMDS)

Input: the coordinates $p_i$ of data items; the region $\Gamma$.
Construct a proximity graph $G$ by Delaunay triangulation.
**repeat**
  Calculate the ideal distance according to (3) for all edges.
  **while** (iteration $< N$) **do**
    Update each $p_i$ according to (6).
  **end while**
  Construct a proximity graph $G$ by Delaunay triangulation.
  Augment $G$ with edges from pairs of nodes that overlap.
**until** (All data items are inside $\Gamma$ with no overlap)
Align the data items horizontally and vertically according to (7).

To avoid introducing overlaps after the alignment, input nodes are scaled so that the total sum of their area is sufficiently less than $A_\Gamma$; otherwise, the CMDS algorithm has to be rerun to remove the overlap, which increases running time and may not generate a desirable layout. For example, if $\Gamma$ is square (with an area of $|\Gamma|$), we can set the area of each node $A_i$ to

$$A_i = \frac{|\Gamma|}{(\sqrt{|V|}+1)^2}. \tag{8}$$

We now discuss the complexity of the CMDS algorithm. A Delaunay triangulation can be found in $O(|V|log|V|)$ time [14]. A scan-line algorithm to find all the overlaps can be implemented in $O(l|V|(\log|V|+l))$ time [10], where $l$ is the number of overlaps. A sweep-line algorithm has a similar time complexity ($O((l+|V|)\log|V|)$) [6]. Because we only apply the scan-line algorithm after no more node overlaps are found along proximity graph edges, $l$ is usually a very small number, hence this step can be considered as having complexity $O(|V|\log|V|)$. Calculating ideal distances takes $O(|E|)$ time, and iteratively solving the proposed model takes $O(N|E|)$ time when edges for each node are pre-stored. Therefore, overall, Algorithm 1 takes $O(c(|V|log|V|+|E|+N|E|))$ time, where $c$ is the total number of iterations in the outer loop of Algorithm 1. In our implementation (described in the next subsection), $N = 20$ is generally enough to achieve acceptable layouts.

### 4.2 Implementation

We implemented the CMDS algorithm in HTML5 and Javascript. A demonstration of CorrelatedMultiples generated by CMDS is presented in a video, at `http://vimeo.com/50263134`. Because we did not find a suitable red-black tree library in Javascript for implementing the scan-line algorithm, we substituted a naïve $O(|V|^2)$ node collision detection algorithm. This is reasonable because a typical computer display of about $1000 \times 1000$ pixels can only fit about 400 objects if we allow each object to be about $50 \times 50$ pixels, so $|V|$ is not that large. For larger $|V|$, a log-linear collision detection algorithm such as a scan-line algorithm would still be desirable.

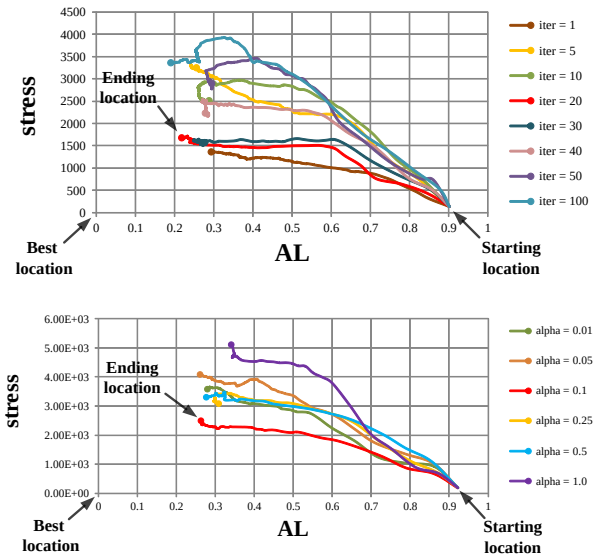


Fig. 3. Stress-AL Pareto curves showing progression of iteration applied to 100 nodes in a $1100 \times 1100$ pixel screen space: (top) for the number of inner iterations $N$ in Algorithm 1, too large or too small values will lead to high stress or poor space utilization; (bottom) for $\alpha$ in (6), large or small values will result in high stress or poor space utilization.

Several additional parameters need to be defined in an implementation, such as the number of iterations $N$ in Algorithm 1, and the parameter $\alpha$ in (6). Our approach is to study the trade-off (Pareto curve) between stress and display-space utilization, and choose parameters that yield the best results. We first define the area loss measure ($AL$)

$$AL = 1 - \frac{\sum_{i \in V} A_i}{B_V + t * \sum_{\{i,j\} \in V} A_i \cap A_j}, \tag{9}$$

where $A_i$ is the area that node $i$ covers, $B_V$ is the bounding box of all nodes, and $t$ is a penalty factor for node overlaps. This penalty must be accounted for, because otherwise a layout where every node sits on the same point would waste the least area. Since overlap removal is essential for small multiples, we set $t = 10$ after testing different values experimentally. The stress with reference to the initial structure is denoted as

$$Stress = \sum_{(i,j) \in E} w_{ij}(s\|p_i - p_j\| - l_{ij})^2, \tag{10}$$

where $w_{ij}$ is a weighting factor, $l_{ij}$ is the initial length of edge $(i, j)$, and $s$ is a scaling factor that minimizes (10)

$$s = \frac{\sum_{(i,j) \in E} w_{ij} l_{ij} \|p_i - p_j\|}{\sum_{(i,j) \in E} w_{ij} \|p_i - p_j\|^2}. \tag{11}$$

For each parameter, our goal is to find a value that achieves good space usage while keeping the stress low.

Given an initial placement, we run CMDS and calculate area loss (9) and stress (10) for different settings of the number of iterations $N$ in Algorithm 1 and $\alpha$ in (6). As shown in Figure 3 (top), when the number of inner iterations $N$ is too small ($N = 1$), space utilization is poor. This is because, without enough iterations of (6), the ideal distance is updated based on a configuration that has not yet converged. Too many inner iterations ($N \geq 40$) might result in higher stress because of

| $\|V\|$ | Time | #Iterations (c) | Time/Iteration |
|---|---|---|---|
| 20 | 0.05 | 32 | 0.001 |
| 40 | 0.05 | 10 | 0.005 |
| 80 | 0.28 | 23 | 0.012 |
| 160 | 1.56 | 40 | 0.039 |
| 240 | 1.78 | 41 | 0.043 |
| 320 | 4.67 | 49 | 0.095 |
| 400 | 6.22 | 44 | 0.141 |

Table 1. CPU time (in seconds) of CMDS algorithm.

doing too much work to satisfy ideal edge length that were determined before the inner iteration starts, and most of the overlaps are usually removed after a few iterations. Excessive iterations without rechecking overlap generally makes layouts that deviate too much from the initial layout, increasing the stress. Therefore, based on Figure 3 (top), we set $N = 20$ to balance stress reduction with area utilization. Figure 3 (bottom) demonstrates the effect of $\alpha$ in (6) with the Pareto curve of stress and area loss. If $\alpha$ is too small ($\alpha = 0.01$), we are essentially solving a conventional sparse stress model without the boundary constraint; on the other hand, if $\alpha$ is large ($\alpha \geq 0.25$), flipping may occur because of abrupt changes of nodes close to the boundary. Experimentally, we found that $\alpha = 0.1$ yields good results.

We applied K-means clustering based on Euclidean distance in the MDS initial layout. Haroz *et al.* [19] recently showed that a compromise must be made between the number of nominal categories and the perceptual complexity of a visualization, and user performance is significantly limited beyond 5 categories. Based on their findings and our experiments, we chose K to be less than 5 for differentiating groups in CorrelatedMultiples. As an alternative, we also experimented with modularity clustering [25] on the similarity graph, which gives comparable clustering in general, and automatically determines the number of clusters. The downside is that with modularity clustering some clusters may become spatially dis-contiguous, due to the uncoupling of the clustering and layout.

### 4.3 Performance

To investigate the performance of CMDS, we tested our implementation in Google Chrome on an i7-2600, 3.4 GHz CPU computer. We started with 20 to 400 nodes at randomly generated positions. The CPU time needed for CMDS to converge is shown in Table 1. As the number of nodes increases, the solver costs more time per iteration. Its running time is dominated by the naïve node collision detection algorithm mentioned earlier, which has $O(|V|^2)$ complexity. Fortunately, in our case $|V|$ is not expected to be large. Moreover, we render the solution process as an animation, and we observe that rendering time dominates the solver's running time, so our solver's performance is satisfactory for a prototype. For larger $|V|$, implementing a scan-line algorithm would speed up the solution process considerably, and reduce its time complexity to $O(c(|V|log|V| + |E| + N|E|))$. (For reference, the PRISM algorithm [15], on which the CMDS solver is based, takes a few seconds to remove overlaps in tens of thousand of nodes.)

## 5 Evaluation

This section describes a controlled user study that shows the effectiveness of CorrelatedMultiples in comparison with the conventional sequential small multiples for searching similar items, as well as a quantitative evaluation that demonstrates how the CMDS method outperforms competing methods in preserving spatial proximity.

### 5.1 User Study

We conducted a controlled experiment to evaluate the effectiveness of CorrelatedMultiples for visual search of similar items, compared with conventional sequential small multiples. We recruited 12 subjects (10 males, 2 females) having background in computer science or software engineering. The subjects ranged in age between 24 and 34 years, with a mean age of 27. Most of the subjects (92%) said they were not already familiar with the notion of small multiples.

#### 5.1.1 Visualizations and Datasets

We collected a data set of CPU usage from 5144 network devices, sampled at 5-minute intervals over one day. We plotted the time series of 288 data points for each device using an area chart, and randomly selected a subset of the charts to make visualizations with conventional small multiples (denoted as **SM**) and CorrelatedMultiples (denoted as **CM**). The charts in **SM** were arranged by device ID in increasing order, while those in **CM** are based on data similarity.

To measure similarity in CPU usage, we applied *Dynamic Time Warping (DTW)*, a technique that non-linearly warps one time series to the other at a minimum cost. Given two time series, DTW finds a warping path which minimizes the total distance between the two series. In our scenario, since the distance between time series should account for the difference in time, we adapted the DTW algorithm by incorporating a locality constraint to align each time step with only its neighboring time steps within a predefined time window $w$ (when $w = 0$, the DTW distance is equivalent to the Euclidean distance). We set $w = 12$ in our experiments, considering two time series to be very similar if they have approximately the same pattern within one hour of time shift. Dynamic programming is then used to find the optimal warping path. Based on the similarity relationship obtained from DTW, we applied CMDS to generate CorrelatedMultiples.

#### 5.1.2 Tasks

Subjects were asked to perform an adapted visual search task. Usually in a conventional visual search task, a subject looks for a specific target from multiple items where there is about 50% of chance that the target is actually present [32]. In our experiments, subjects were asked to identify one or multiple items that are the most similar to a target given in the visualization. This task represents a typical visual search task that involves visual comparison.

For both visualizations, we created 10 datasets with varying number of charts (from 50 to 150). Figure 4 shows a configuration of 125 items for both visualizations, in which the given target is highlighted by a rectangle in a distinct color.

#### 5.1.3 Procedure

The study was conducted as a within-subjects experiment with 2 experimental conditions (**SM** or **CM**) and 10 repetitions (visualization image) for each condition. For each repetition, the subject was presented with only one condition. We counter-balanced the selection of condition in the 10 repetitions so that each subject performed one repetition for both conditions with the same number of charts. For example, if a subject accomplished one task for **SM** with 100 items, he would also be asked to perform another task for **CM** with 100 items. The order of tasks was random.

The study was performed on an i7-2600, 3.4 GHz CPU desktop computer equipped with a standard 24-inch screen of resolution of $1920 \times 1080$ pixels. Prior to the experiment, the subjects viewed a tutorial that provided a basic explanation of small multiples, and they performed some training tasks to get familiar with the user interface of the experimental system. For each task, the subject was given a randomly chosen **SM** or **CM** visualization, and prompted to answer the question *Find 'i' most similar area chart(s) to the highlighted one* ($i = 1, 2$ or 3 randomly). After typing in the answer(s) and clicking on the "next' button, the next task was loaded. After the subjects finished all tasks, they were asked to rate their satisfaction with CorrelatedMultiples on a questionnaire containing 6 questions, and finally, to participate in a semi-structured interview.

#### 5.1.4 Hypotheses

The major goal of the user study is to assess the effectiveness of CorrelatedMultiples for visual search of similar items. We formulated the following hypotheses for this study:

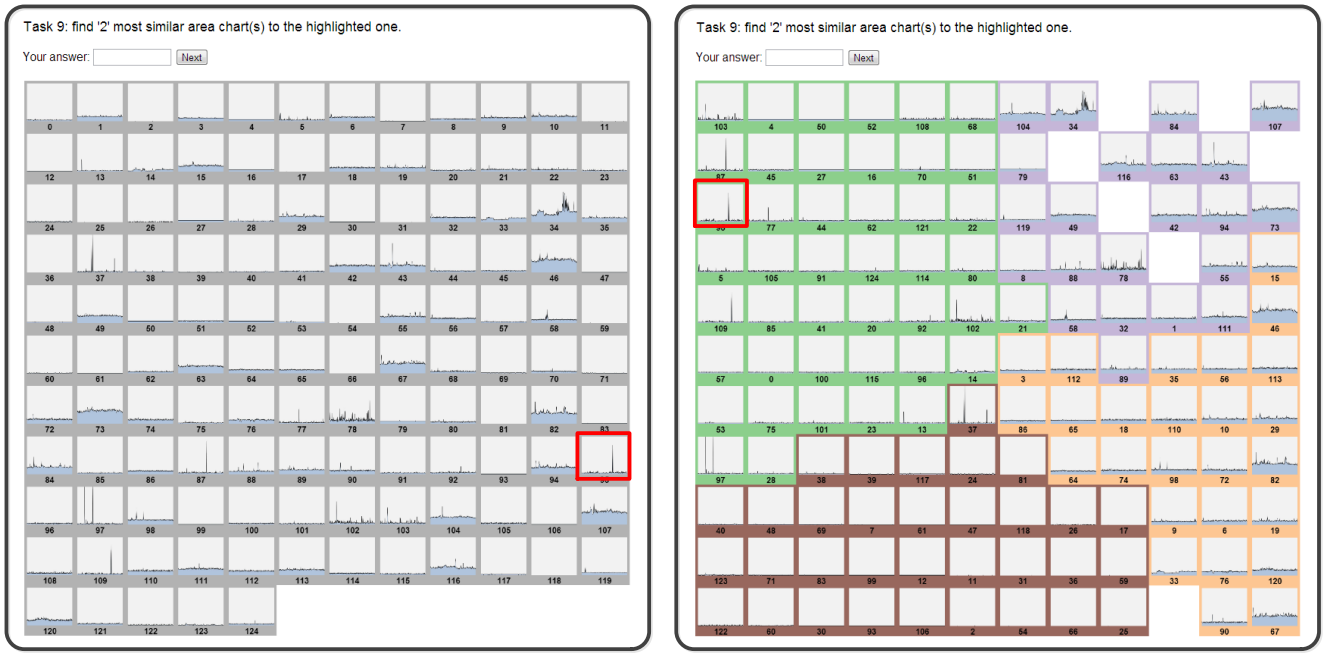


Fig. 4. Time series charts used in the user study system. Left: Sequential small multiples (SM). Right: CorrelatedMultiples (CM). In each task a subject was presented with either a SM or a CM and asked to choose 1 (or 2, 3) similar chart(s) to the highlighted one. The user study system is presented in a video, at `http://vimeo.com/50263134`.

**[H1]** *CorrelatedMultiples support better performance than conventional small multiples.* Because spatial proximity is important for visually searching for similar items, we expect CorrelatedMultiples to outperform conventional small multiples.

**[H2]** *Sequential small multiples do not have a significant negative impact on accuracy compared with CorrelatedMultiples.* Although we believe that the conventional small multiples require more time for visual search tasks, we hypothesize they do not negatively affect visual comparison once the items to compare are found.

**[H3]** *CorrelatedMultiples have a positive impact on user satisfaction.* Because spatial proximity provides a strong cue for search, we hypothesize that users prefer to group similar items spatially.

### 5.1.5 Results and Discussion

We measured the completion time the subjects needed to find items similar to a given target, the accuracy of their selections, and the subjective assessment from the evaluation questionnaires. Task completion time and accuracy measures were evaluated using single factor Analysis of Variance (ANOVA) for the dependent variables. We found a significant main effect for task completion time ($F(1,22) = 4.8668, p = 0.01427$). The average time spent on a task was 35.67 seconds for **SM**, and 20.46 seconds for **CM**, as illustrated in Figure 5 (left). Task accuracy was not found to have a significant difference ($F(1,22) = 0.5417, p = 0.3239$). On average, the accuracy of a task was 53.3% for **SM**, and 60.0% for **CM**, as shown in Figure 5 (right).

The questionnaire asked subjects to assess their satisfaction with **CM** against **SM** on multiple criteria: ease (*CorrelatedMultiples made it easier in answering the questions*), efficiency (*I could find the similar items more quickly with CorrelatedMultiples*), confidence (*I was more confident in my answer with CorrelatedMultiples*) and aesthetics (*CorrelatedMultiples is visually more pleasing*). The score scale is from 1 (totally disagree) to 5 (totally agree), with 3 as neutral. Figure 6 provides the average ratings for each criterion. Generally **CM** is rated higher than **SM**, in particular the subjects were more efficient and confident in the visual search tasks with **CM** than with **SM**.

In the interview, two subjects mentioned that although it was easier to accomplish tasks with **CM**, they still did a one-time linear scan

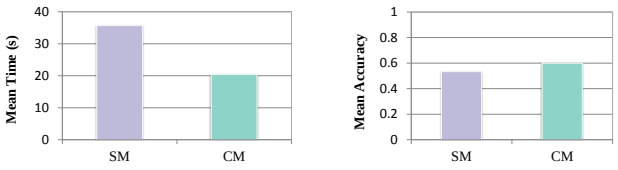


Fig. 5. Mean task completion time (left) and accuracy (right) for the user study. Average time spent on a task was 35.67 seconds for conventional small multiples (SM), and 20.46 seconds for CorrelatedMultiples (CM). Average accuracy on a task was 53.3% for SM, and 60.0% for CM.

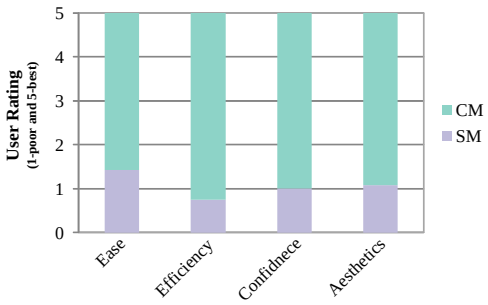


Fig. 6. Subjective satisfaction of CorrelatedMultiples (SM) against conventional small multiples (SM) on the criteria of *easy*, *efficiency*, *confidence* and *aesthetics*. The scale is from 1 (poor) to 5 (best).

(just as they did with **SM**) to be sure that they found the right answer. Four subjects reported that they were not aware of the color encoding for the groups in **CM** at first, but were still able to find similar items

more quickly and confidently since they were placed nearby (due to the CMDS layout). Others expressed that they figured out the function of color encoding as they got more familiar with **CM**, and it helped them in later tasks. This suggests that **CM** could potentially enhance the performance further for visual search tasks if the subjects were explicitly made aware of the color encoding scheme. The feedback on **CM** obtained in the interviews was generally positive. Eight subjects mentioned that **CM** was "helpful", "beneficial" or "useful".

Based on these results, the hypotheses are well supported. The shorter task completion time and the (mostly) higher subjective assessment of efficiency supports **H1**. **H2** is supported because even though most subjects received relatively fewer correct results with **SM** and felt they are less confident in their answers with **SM**, the difference in the task accuracy is not statistically significant. On the other hand, this also indicates that **CM** does not increase the perceived complexity of the visual search task by associating categories with colors. **H3** is supported by the questionnaire results, which indicate that most subjects preferred the spatial coherence provided by **CM**.

In summary, the user studies show that CorrelatedMultiples are useful in helping users visually identify similar items, and they can perform this more quickly. Encoding spatial coherency within small multiples was valued by the users. Most users preferred CorrelatedMultiples.

### 5.2 Quantitative Evaluation

While CorrelatedMultiples outperform conventional small multiples for visual search, it is still important to know how well they preserve spatial proximity in a quantitative manner. Therefore, we performed a quantitative evaluation of CMDS in comparison with two state-of-the-art grid layout methods: SpatialGrid (spatially ordered Treemap [33]) and GridMap [13].

#### 5.2.1 Existing Methods

**SpatialGrid**, which is adapted from spatially ordered Treemap proposed by Wood and Dykes [33], produces a grid layout in which grid cells are ordered with two-dimensional consistency [13]. Based on the Squarified Treemap layout algorithm [5], SpatialGrid recursively processes a single row or column (the shorter one) of the grid cells. Specifically, given an initial layout $I$ and a target grid $G$, SpatialGrid first finds the point in $G$ that is closest in Euclidean distance to its position in $I$, for each cell in the shorter side of the grid. The remaining grid cells in $G$ form a sub-grid that is exactly one row or column smaller than before, which is processed recursively. For more details we refer to [33] and [13].

**GridMap**, proposed by Eppstein *et al.* [13], computes an optimal point set matching between the regions of an original geographic map to the cells of a grid. Given an initial geographic map $M$ and a target grid $G$ of the same number of points in the plane, GridMap computes an optimal one-to-one matching between $M$ and $G$ with a translation and/or scaling that minimizes the total distance. Basically it checks all possible cases of matching with translations and/or scalings, and for each case it solves the underlying point set matching problem via linear integer programming. The optimal matching is the one with the minimal total displacement. Hereafter, we refer to the $L_1$ method detailed by Eppstein *et al.* [13] that minimizes the displacement with translations as the *TransGridMap* method, and the $I$ method that minimizes displacement without translation or scaling as the *SimpleGridMap* method.

#### 5.2.2 Evaluation Metrics

To compare the quality of the above methods with CMDS, we employed a set of metrics derived from [33] and [13] — the ratio of displacement, the percentage of recalled adjacency and the percentage of the preserved directional relation. We also measured the computational performance of each method.

The *ratio of displacement* is the average Euclidean distance by which nodes have been displaced, and is scaled between 0 (no displacement) and 1 (maximum possible displacement). In equation (1) proposed by Wood and Dykes [33], the ratio of displacement is formulated as:

$$disp = \frac{\sum d_i}{|V|\sqrt{A_{root}}}, \tag{12}$$

where $d_i$ is the Euclidean distance between each cell in the initial layout $I$ and the target grid $G$, $|V|$ is the number of nodes and $A_{root}$ is the area of the bounding area of $I$ (also the same bounding area of $G$).

Recall measures the fraction of relevant instances retrieved [26]. In our case, the relevant instances are the adjacent regions in the original layout $I$, while the retrieved instances are the adjacent cells in the resulting grid $G$. For each cell in $G$, its *recalled adjacency* is defined as the ratio between the number of items that are both adjacent in $I$ and $G$, and the number of adjacent items in $I$. Following the criterion proposed by Eppstein *et al* [13], two regions or grid cells are considered adjacent if the intersection of their closed boundaries is nonempty, suggesting that each grid cell has at most eight neighbors.

The *preserved directional relation* measures the number of pairs of cells that have the same orthogonal ordering in both the original layout $I$ and the resulting grid $G$. The number of such pairs $P$ is defined as:

$$P = |(x,y)|(x,y) \in I \times I \ \wedge \ o(I,x,y) = o(G,x,y)|, \tag{13}$$

where $o(I,x,y)$ is the orthogonal order of $(x, y)$ in $I$ (analogously for $o(G,x,y)$). The number is normalized by the total number of possible pairs to give a percentage.

To compare the computational performance of these two previously published methods with CMDS, we implemented them in Javascript and HTML5. The optimal point set matching for the grid map was achieved using the integer linear programming (ILP) based on a GNU Linear Programming Kit for Javascript (GLPKJS) [17]. We ran each method on a geographical map of United States (from Eppstein *et al* [13]) in a Google Chrome browser on the same experimental machine as in Section 4.3. To be consistent with the measurement by Eppstein *et al* [13], we considered only the 48 contiguous states as the initial layout, and the target layout as an $8 \times 6$ grid.

#### 5.2.3 Results and Discussion

Table 2 shows the results of the measurement. We can observe that CMDS recalled more adjacent items than SpatialGrid and TransGridMap, with the same recall as SimpleGridMap. CMDS also preserved more directional relations than SpatialGrid and SimpleGridMap, but a bit less than TransGridMap. While TransGridMap performed best in minimizing total displacement, this was not a surprise since the method is designed to minimize this quantity. We can see that CMDS was still better than SpatialGrid (32.9% improvement) and SimpleGridMap (27.8% improvement). In terms of running time, SpatialGrid took the least time. CMDS was roughly 3.2 times faster than SimpleGridMap, and about 150,000 times faster than TransGridMap. This is expected since TransGridMap requires the computation of many $(48 \times 48 \times 48)$ distance matchings [13].

The corresponding grid layouts in Figure 7 confirmed these quantitative results. Because SpatialGrid processed a single row or column of the grid cells starting from the top left corner in a greedy manner, some cells were placed relatively far from their original positions. For instance, the group of cells IL, OH, WI, MI, which are located in the middle right of the initial map, were placed in the middle top of SpatialGrid. As for SimpleTransGrid, cells such as FL, CO and AZ were placed relatively far from their original positions. Cells in TransGridMap and the CMDS layout seemed spatially stable compared with the other two methods.

Overall, CMDS outperformed SpatialGrid and SimpleGridMap in minimizing displacement, recalled more adjacent items than SpatialGrid and TransGridMap, better preserved the directional relation than SpatialGrid and SimpleGridMap, and is more computationally efficient than SimpleGridMap and TransGridMap.

## 6 Applications

Next we will discuss application scenarios and describe several interesting findings. We studied the effectiveness of CorrelatedMultiples

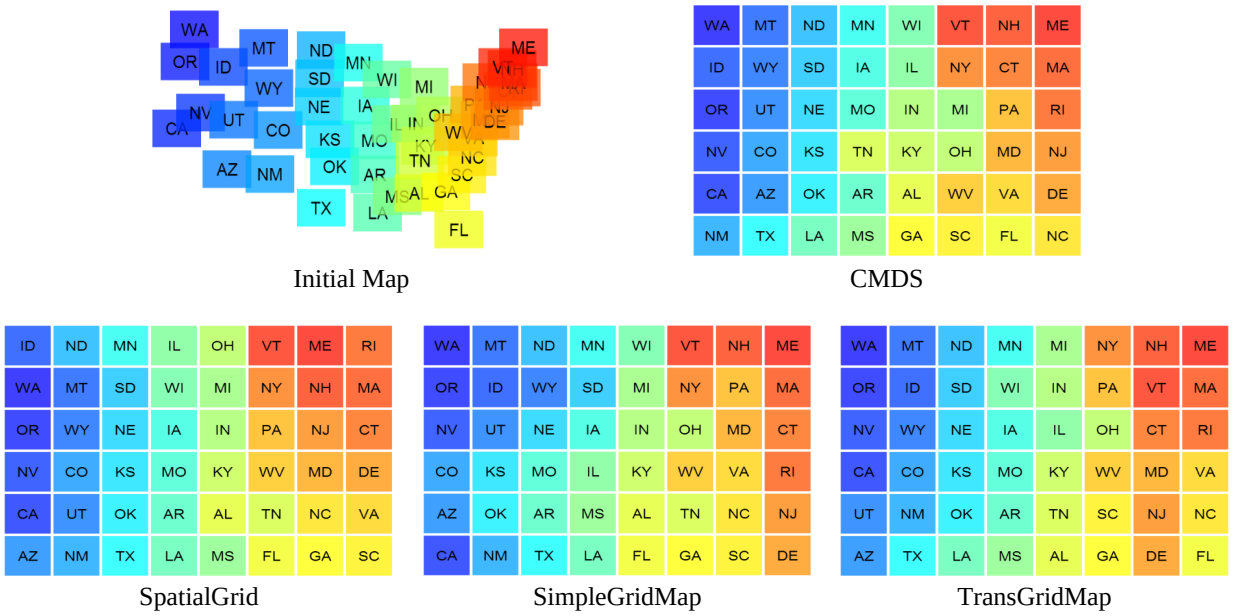


Fig. 7. Grid layouts of USA map. The initial map consists of the 48 contiguous states, from which the generated grid layouts are: CMDS, SpatialGrid, SimpleGridMap, and TransGridMap, respectively.

| Method | Displacement | Adjacency Recall | Directional Relation | CPU Time(s) |
|---|---|---|---|---|
| CMDS | 0.1811 | 75.24% | 89.80% | 0.162 |
| SpatialGrid | 0.2697 | 73.33% | 89.01% | 0.010 |
| SimpleGridMap | 0.2508 | 75.24% | 89.72% | 0.521 |
| TransGridMap | 0.1767 | 74.29% | 90.69% | 23998.484 |

Table 2. Numerical measures of the grid layouts in Figure 7. CMDS outperformed SpatialGrid and SimpleGridMap in minimizing displacement, recalled more adjacent items than SpatialGrid and TransGridMap, better preserved the directional relation than SpatialGrid and SimpleGridMap, and is more computationally efficient than SimpleGridMap and TransGridMap.

in two applications: stock market trend analysis, and Madden-Julian oscillation in climate modeling.

## 6.1 Stock Market Trend Analysis

### 6.1.1 Application Scenario

The Dow Jones Industrial Average (DJIA) is one of the most widely followed stock market indices. It has been tracked since May 26, 1896, and shows how the stocks of 30 large publicly-owned U.S. companies are traded. Because the DJIA represents some of the largest companies, large fluctuations can indicate losses and potential pullbacks in the U.S. economy (shown in Figure 8 (left)), or signify real or anticipated high levels of growth and profits (shown in Figure 8 (right)). While studying historical DJIA trends and variations can help individuals and companies to make investment decisions, the patterns are often too complicated to remember and compare since trends can vary widely. Understanding similarities and contrasts in annual trends over the lifetime of the index may help analysts to discern overall patterns.

### 6.1.2 Dataset

We studied DJIA trends by year, from 1897 to 2011. Each year contains about 250 time steps (all weekdays, excluding holidays). Since we are interested in relative fluctuations in the DJIA, for each year we placed the beginning of the year at the origin, and plot the percentage of change at each time step. Examples of DJIA trend charts are shown in Figure 8.

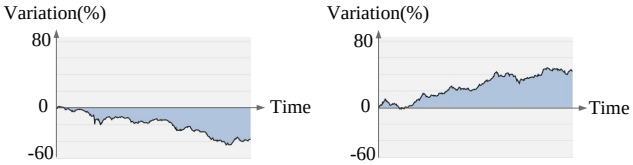


Fig. 8. The Dow Jones Industrial Average (DJIA) trends: (left) a falling trend can indicate losses and potential declines in the U.S. economy; (right) a growing trend may signify high levels of growth and profits.

### 6.1.3 Visual Analysis

To measure similarity in the DJIA time series between two years, we use the DTW-based similarity measure described in Section 5.1.1. Based on the similarity relationship obtained from DTW, we applied CMDS to render CorrelatedMultiples for DJIA trends. Each of the yearly trends was depicted as a time-series chart, labeled by year. Figure 1 shows two different visualizations of the DJIA over 115 years. In the sequential small multiples (left), charts were arranged in the increasing order of year, while the charts in CorrelatedMultiples (right) were placed in a spatially coherent manner.

In the CorrelatedMultiples of Figure 1 (right), we can see three large clusters, which can be identified as: falling trends, stable trends, and rising trends (from top to bottom). In general, the closer to the top in the figure, the more the DJIA fell over time in the corresponding year. Near the bottom of the figure, we see increases in the index over a year. As expected, stable trends are placed in the middle of the figure,

between the rising and falling trends. Because of the spatial proximity that the CorrelatedMultiples preserve, trends of similar fluctuations are placed nearby, which helps us identify similar patterns. For example, most of the trends at the top of the figure in the CorrelatedMultiples reflect severe economic crises, such as in 2008, 1907 and 1931 (from left to right in the top row), all of which saw sharp declines (around 50% during the year). These correspond to *the Global Financial Crisis*, *the 1907 Bankers' Panic* and *the Great Depression*, respectively. Interestingly, a dramatic rising trend of the year 1933 in the bottom-left of the CorrelatedMultiples (which is the farthest from the year 1931 in the visualization) represents a great recovery from the Great Depression — more than 75% of the nation's banks had been reopened and several acts were passed by the U.S. Congress to revive the economy in that year. With CorrelatedMuliples, we are able to quickly identify similar and dissimilar trends, based on spatial locations.

### 6.2 Madden-Julian Climate Modeling

#### 6.2.1 Application Scenario

Madden-Julian Oscillation (MJO) is a phenomenon manifested as intra-annual weather fluctuation in the tropics, mainly in the Indian and western Pacific Oceans. It is considered by some to be a key mechanism explaining weather patterns in those regions, and involves relationships between the atmosphere and ocean currents. It is often characterized by an eastward progression of enhanced and suppressed tropical rainfall. Analyzing water vapor intensity over time and a range of longitudes allows one to gain better understanding about how MJO behaves over time. Figure 9 shows two examples of water vapor distribution, visualized as area charts. The middle region has higher water vapor in Figure 9 (left), while the water vapor in Figure 9 (right) generally increases as it moves eastward. However, as the number of time steps increases, it becomes more difficult to examine individual distributions. Moreover, since MJO is periodic, peak water vapor may occur in the same area at some previous time steps. Visual comparison can be improved if similar distributions at different time steps are placed together, to enable better understanding of periodic patterns.

#### 6.2.2 Dataset

We applied CorrelatedMultiples to an MJO simulation performed by the Pacific Northwest National Laboratory [18]. The data set made by this simulation consists of 479 time steps, recorded at 6 hour intervals (from October 1, 2007 to January 29, 2008). We used the time-varying water vapor intensity collected in the region of $[60°E - 150°E]$ over time as the raw data, and sampled it every 4 time steps to obtain 119 days of input data. For each selected time step, we computed the distribution of water vapor by longitude in the given region, and rendered the distribution as an area chart (as shown in Figure 9).

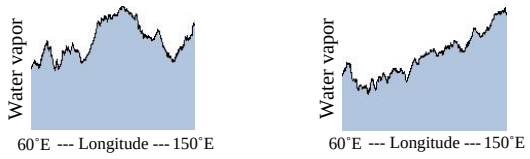


Fig. 9. Aggregated distributions of water vapor mixing ratios in the Madden-Julian Oscillation (MJO) simulation in the region of $[60°E - 150°E]$ at two different time steps: (left) the middle and last region have higher water vapor; (right) the peak of the water vapor has moved eastward.

#### 6.2.3 Visual Analysis

Since scientists are generally interested in studying time-varying patterns of MJO, such as shifts in the peaks, we normalized the water vapor distribution at the selected time step, and computed Earth Mover's Distance (EMD) [28] to measure the similarity of the distributions between each pair of the selected time steps. EMD assigns a large dissimilarity to two time steps if their peaks are far apart. We generated

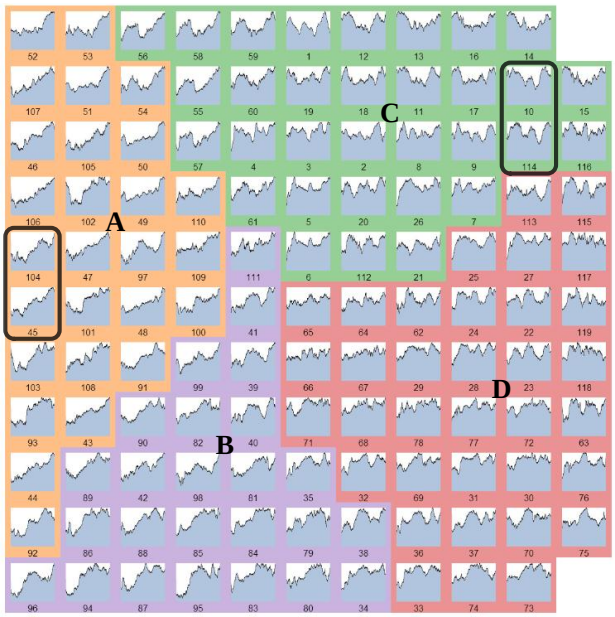


Fig. 10. Correlated water vapor distributions in the Madden-Julian Oscillation (MJO) simulation visualized by CorrelatedMultiples. Each area chart represents one day of simulation. The area charts are divided into four clusters based on their similarity measured by Earth Mover's Distance (EMD). Day 45 and 104, and day 10 and 114 are highlighted to imply periodic patterns during the MJO evolution since they are similar but with a relatively large time interval.

CorrelatedMultiples using this dissimilarity measure. Each distribution was rendered using an area chart labeled by its time step, and the ideal distance $l_{ij}$ in (3) of every pair of distributions was set to the EMD value.

Figure 10 shows CorrelatedMultiples of water vapor distributions from the MJO simulation. Time steps in the same cluster have similar distributions. Since rainfall oscillation is reflected in the shape of water vapor intensity, we can observe some interesting patterns from the top-right to the bottom-left of Figure 10: in cluster **A**, peaks generally showed up on the right, which means tropical rainfall reached its max in the east side of the given region; while in cluster **B**, most of the peaks appeared in the middle of the charts; in cluster **C**, more peaks appeared at the west and east sides of the given region; and in cluster **D**, most charts have multiple peaks, evenly distributed, which suggests the MJO occurred more frequently in the corresponding time steps of this cluster. Furthermore, it is apparent from the figure that some nearby charts with a large time interval have similar distributions, such as day 45 and 104, and day 10 and 114 (highlighted in the figure). This seems to imply a periodic pattern as MJO progresses.

In summary, by means of CorrelatedMultiples, we were are to identify time steps with similar distributions, and observe certain periodic behavior, without having to compare all pairs of time steps. The spatial proximity preserved by CorrelatedMultiples also allows comparing and analyzing distributions at the cluster level.

## 7 Conclusion and Future Work

We described CorrelatedMultiples, a spatially coherent small multiples visualization. CorrelatedMultiples retain the major advantages of small multiples to allow side-by-side visual comparison, but also exhibit spatial locality based on similarity in the data. Layouts are made by a novel Constrained Multi-Dimensional Scaling (CMDS) algorithm. We explored the effectiveness of CorrelatedMultiples over sequential small multiples in a controlled user study, and quantitatively evaluated the quality and performance of the CMDS method compared

with previous methods. We showed the benefits of CorrelatedMultiples to applications in stock market trend analysis and Madden-Julian oscillation climate analysis.

Although in this paper, CMDS was applied only to arrange items in rectangular spaces, the algorithm can be applied to regions with other shapes, such as circles or the shape of a country's geographical boundary. In the future, we plan to adapt CMDS to dynamic visualization of small multiples where we want to simultaneously maintain layout stability and maximize screen space utilization.

## REFERENCES


[1] D. Archambault, H. Purchase, and B. Pinaud. Animation, small multiples, and the effect of mental map preservation in dynamic graphs. *IEEE Transactions on Visualization and Computer Graphics*, 17(4):539–552, Apr. 2011.

[2] M. Baur and U. Brandes. Multi-circular layout of micro/macro graphs. In *Proceedings of the 14th International Conference on Graph drawing*, volume 4875, pages 255–267, 2007.

[3] B. B. Bederson. Photomesa: a zoomable image browser using quantum treemaps and bubblemaps. In *UIST'01*, pages 71–80, 2001.

[4] J. Bertin. *Graphics and graphic information-processing*. de Gruyter, 1981.

[5] M. Bruls, K. Huizing, and J. van Wijk. Squarified treemaps. In *In Proceedings of the Joint Eurographics and IEEE TCVG Symposium on Visualization*, pages 33–42, 1999.

[6] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 3rd edition, Apr. 2008.

[7] T. Dwyer. Scalable, versatile and simple constrained graph layout. *Computer Graphics Forum*, 28(3):991–998, 2009.

[8] T. Dwyer and Y. Koren. Dig-cola: Directed graph layout through constrained energy minimization. In *Proceedings of the 2005 IEEE Symposium on Information Visualization*, pages 65–72, 2005.

[9] T. Dwyer, Y. Koren, and K. Marriott. Constrained graph layout by stress majorization and gradient projection. *Discrete Mathematics*, 309(7):1895–1908, 2009.

[10] T. Dwyer, K. Marriott, and P. J. Stuckey. Fast node overlap removal. In *Proc. 13th Intl. Symp. Graph Drawing (GD '05)*, volume 3843 of *LNCS*, pages 153–164. Springer, 2006.

[11] T. Dwyer, K. Marriott, and M. Wybrow. Dunnart: A constraint-based network diagram authoring tool. In *Proceedings of the 16th International Conference on Graph Drawing*, GD'08, pages 420–431, 2008.

[12] T. Dwyer and G. Robertson. Layout with circular and other non-linear constraints using procrustes projection. In *Proceedings of the 17th International Conference on Graph Drawing*, volume 5849/2010 of *GD'09*, pages 393–404, 2009.

[13] D. Eppstein, M. van Kreveld, B. Speckmann, and F. Staals. Improved grid map layout by point set matching. In *PacificVis*. IEEE, 2013.

[14] S. Fortune. A sweepline algorithm for voronoi diagrams. In *Proceedings of the second annual Symposium on Computational geometry*, SCG '86, pages 313–322, New York, NY, USA, 1986. ACM.

[15] E. R. Gansner and Y. Hu. Efficient node overlap removal using a proximity stress model. In *Graph Drawing*, pages 206–217. Springer-Verlag, Berlin, Heidelberg, 2009.

[16] E. R. Gansner, Y. Koren, and S. North. Graph drawing by stress majorization. In *Proceedings of the 12th International Conference on Graph Drawing*, GD'04, pages 239–250, 2004.

[17] GlpkJS. A GNU linear programming kit for Javascript, http://hgourvest.github.com/glpk.js/.

[18] S. Hagos and L. R. Leung. Moist thermodynamics of the madden-julian oscillation in a cloud-resolving simulation. *Journal of the Atmospheric Sciences*, 24(21), 2011.

[19] S. Haroz and D. Whitney. How capacity limits of attention influence information visualization effectiveness. *IEEE Trans. Vis. Comput. Graph.*, 18(12):2402–2410, 2012.

[20] T. Itoh, C. Muelder, K.-L. Ma, and J. Sese. A hybrid space-filling and force-directed layout method for visualizing multiple-category graphs. In *Proceedings of the 2009 IEEE Pacific Visualization Symposium*, pages 121–128. IEEE Computer Society, 2009.

[21] W. Javed, B. McDonnel, and N. Elmqvist. Graphical perception of multiple time series. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):927–934, Nov. 2010.

[22] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31:7–15, 1989.

[23] J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29:1–27, 1964.

[24] J. Marks, K. Ryall, and S. Shieber. An interactive constraint-based system for drawing graphs. In *Proceedings of the ACM Symposium on User interface software and technology*, pages 97–104, 1997.

[25] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, June 2006.

[26] V. Raghavan, P. Bollmann, and G. S. Jung. A critical investigation of recall and precision as measures of retrieval system performance. *ACM Trans. Inf. Syst.*, 7(3):205–229, July 1989.

[27] G. Robertson, R. Fernandez, D. Fisher, B. Lee, and J. Stasko. Effectiveness of animation in trend visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1325–1332, Nov. 2008.

[28] Y. Rubner, C. Tomasi, and L. J. Guibas. A metric for distributions with applications to image databases. In *Proceedings of the Sixth International Conference on Computer Vision*. IEEE Computer Society, 1998.

[29] B. Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Trans. Graph.*, 11(1):92–99, Jan. 1992.

[30] R. Tamassia. Constraints in graph drawing algorithms. *Constraints*, 3(1):87–120, 1998.

[31] E. R. Tufte. *Envisioning Information*. Graphic Press, 1990.

[32] J. M. Wolfe. Visual search. In H. Pashler, editor, *Attention*. University College London Press, London, UK, 1996.

[33] J. Wood and J. Dykes. Spatially ordered treemaps. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1348–1355, 2008.

[34] J. Wood, A. Slingsby, and J. Dykes. Visualizing the dynamics of London's bicycle-hire scheme. *Cartographica*, 46(4):239–251, 2011.