Let It Flow: a Static Method for Exploring Dynamic Graphs

Weiwei Cui* Microsoft Research
Xiting Wang† Tsinghua University
Shixia Liu‡ Microsoft Research
Nathalie H. Riche§ Microsoft Research
Tara M. Madhyastha¶ University of Washington
Kwan-Liu Ma‖ University of California, Davis
Baining Guo** Microsoft Research

ABSTRACT

Research into social network analysis has shown that graph metrics, such as degree and closeness, are often used to summarize structural changes in a dynamic graph. However there have been few visual analytics approaches that have been proposed to help analysts study graph evolutions in the context of graph metrics. In this paper, we present a novel approach, called *GraphFlow*, to visualize dynamic graphs. In contrast to previous approaches that provide users with an animated visualization, *GraphFlow* offers a static flow visualization that summarizes the graph metrics of the entire graph and its evolution over time. Our solution supports the discovery of high-level patterns that are difficult to identify in an animation or in individual static representations. In addition, *GraphFlow* provides users with a set of interactions to create filtered views. These views allow users to investigate why a particular pattern has occurred. We showcase the versatility of *GraphFlow* using two different datasets and describe how it can help users gain insights into complex dynamic graphs.

Keywords: Dynamic Graphs, Flow Visualization.

1 INTRODUCTION

Many graphs, such as the online social network Twitter, are constantly evolving as nodes join, leave, connect to other nodes, and/or disconnect from some of their adjacent nodes. We refer to such graphs as dynamic graphs. They are featured in a wide range of applications from examining how proteins interact to understanding how information flows in large social networks over time. For this reason, there is a great need to analyze an evolving graph.

One major challenge for analyzing dynamic graphs is capturing all the structural changes for a group of nodes while maintaining an overview of the entire evolving graph so that users can easily extract and integrate information across the changing states of the graph [11, 24]. However, existing popular approaches may be inadequate for this task. For example, it is difficult for users to perceive high-level patterns in an animated representation, since research has indicated that it is difficult for people to follow more than six or seven animated elements at the same time [8, 23].

Research into social network analysis has shown that graph metrics are very useful for summarizing structural changes in a dynamic graph [33]. Different metrics emphasize different aspects of a social network. For example, the node degree is usually used to indicate people's engagement with a social network. Illustrating the changes in node degree can reveal high-level patterns, such as the increased/decreased engagement of one community. Similarly, the clustering coefficient can provide an overall indication of the clustering in a network. Furthermore, it is often desirable to understand the relative metric changes of a set of nodes/edges within the context of the whole graph. For example, analysts are concerned with the question of whether the decreased engagement of one community is caused by the overall engagement decline of the social network or by an increased engagement in another related community. Therefore, it is important to take rank factor into account when using graph metrics to summarize a dynamic graph.

In this paper, we introduce *GraphFlow*, a new toolkit for examining and analyzing dynamic graphs from a summary of metric changes to detailed structural changes. We model the metric changes of the nodes/edges in a dynamic graph into a vector field.

Visualizing this vector field provides an overview of the graph, with which users can observe at a glance how a graph changes over time as well as compare the relative metric changes of two or multiple nodes. In addition, we propose an energy-based method to quantitatively measure the changes, so that users can easily identify critical sections of the evolving graph. Furthermore, using a set of well-aligned node-link diagrams, *GraphFlow* enables users to compare detailed graph structures at different time points.

To summarize, *GraphFlow* shows a dynamic graph in a static manner with both high-level insights into the structural changes across several time points and low-level details to investigate and understand these changes. It makes the following contributions:

- **A flow-based visual metaphor** that provides an overview of property changes in a dynamic graph. This visualization supports the discovery of salient features of the dynamic graph, as well as patterns of interest.
- **An energy-based trend analysis** that helps users identify critical time points in the flow visualization. Inspired by the seam carving technique [2], *GraphFlow* uses an energy function to define the importance of time points. In addition to finding critical time points, it can also be used to remove time points with smooth changes. In this way, screen space can be better allocated to the more important information.
- **An interactive, multi-view system** that allows users to analyze a dynamic graph from the global evolving patterns to detailed structural changes over time.

2 RELATED WORK

Much research has explored static graphs [3, 17, 18, 31] over the past two decades. However, the visualization of dynamic graphs is much more challenging and relatively few studies have been conducted in this area. The most well-known approach is the use of an animated node-link diagram to convey the evolution of the graph. Early work on the topic labeled the technique as dynamic graph drawing [9, 16, 15, 21]. The technique consists of generating a sequence of graphs for each time point and animating the layout from one step to the next to help the viewer easily follow changes such as fading in and out of nodes and edges as they appear in or disappear from the diagram.

Although animation techniques are enjoyable and exciting [27], it is challenging to discover certain patterns due to the temporal nature of the animation. For example, it is difficult to remember previously visited states of the graph or compare them if they do not appear

*e-mail:weiwei.cui@microsoft.com
†e-mail:v-xitwan@microsoft.com
‡e-mail:shixia.liu@microsoft.com
§e-mail:nathalie.henry@microsoft.com
¶e-mail:tara.madhyastha@gmail.com
‖e-mail:ma@cs.ucdavis.edu
**e-mail:bainguo@microsoft.com

consecutively in the animation [6]. In addition, high-level patterns often occur over a longer period of time and the maintenance of a mental map may prove cognitively demanding [1]. A solution to tackling this problem is to display all the static graphs per time point. Since this approach requires a large amount of display space, it often uses a set of stacked layers in 2.5D or 3D [6]. Unfortunately, stacking the static graphs together often introduces additional visual clutter and does not scale well for graphs with a large number of time points, nodes, and edges.

To address this problem, recent research has investigated alternative static representations. One technique is EdgeSplatting [11]. It hierarchically organizes vertices of the graphs on vertical, parallel lines that are placed perpendicular to the horizontal time line. Intuitively, the node-link structures of individual graphs are encoded into the texture between neighboring vertex lines. Due to the extra space required to display the texture, EdgeSplattng does not scale well with a large number of time points. Thus, the authors subsequently proposed a "sliding window" approach to solve the scalability issue [4]. A similar pixel-based approach [10] removed the texture between vertex lines to support better scalability in terms of graph size and time. In both approaches, the order of vertices in the vertex lines does not change over time, which helps users to easily track individual vertices. More recently, Sallaberry et al. [29] combined a new evolving clustering algorithm with two visualization techniques for exploring large dynamic graphs. Although these methods have achieved some success in helping users understand the evolution patterns in dynamic graphs, they may fail to discover some patterns related to the graph metric, such as a slow increase in the degree of a subgraph over time.

A few researchers have also investigated the use of matrix representations [7], placing a bar chart or glyph in the cells of the matrix to indicate the evolution of the relationships. These techniques appear very promising as they do not require users to remember graph states at different time points (i.e., all the information is available in the static representation) and can scale the amount of information represented in a single view. However, it remains difficult to extract high-level patterns, such as an overall evolution in the degree of a group of nodes, from the evolving graph.

Recently, a number of approaches have been introduced to extract high-level patterns from graphs. PivotGraph [37] and HoneyComb [35] aggregate a graph based on its data attributes, supporting the discovery of relationships between node attributes. GraphPrism [20] uses graph-theoretic properties such as the diameter to characterize the structure of large networks. However, none of these approaches has been applied to the analysis of dynamic graphs.

Rosvall and Bergstrom [28] extended ThemeRiver [12, 28, 30] to visualize dynamic networks. They split/merged color stripes to represent the splitting/merging patterns between clusters in a network. In contrast, *GraphFlow* does not require cluster information. It focuses on the order of individual nodes and uses the order changes to represent the changes in activeness of graph nodes over time.

GraphFlow aims to characterize the overall structural changes of a dynamic graph by offering a static representation of a number of high-level graph properties. It provides a flow-based visualization for summarizing the overall evolution patterns, a trend analysis for identifying the critical time points in the flow visualization, and a detailed view of the structural changes across several correlated states for investigating the major causes of such patterns.

3 GraphFlow

A dynamic graph can be represented by a sequence of timeslices: $\Gamma = \{G_1, G_2, \ldots, G_n\}$. $G_i = (V_i, E_i)$ ($1 \leq i \leq n$) is a timeslice that encodes the structure of the graph at time i.

To visually convey the evolution patterns in Γ in a static way and within the context of the structure. A straightforward method is to show each of the timeslices in the form of node-link diagrams, from G_1 to G_n, side by side. However, this is not practical when n becomes very large. To solve this problem, we present a visualization framework, *GraphFlow* that allows users to examine a dynamic graph at different levels of detail. In *GraphFlow*, various graph metrics can be used to summarize the structural evolution of a dynamic graph. Once high-level insights are derived using the metrics, detailed graph structures can then be retrieved on demand to help users investigate and understand those insights.

Accordingly, *GraphFlow* consists of two views: 1) a flow view providing a visual summary of Γ to help users understand the overall evolution patterns and identify critical timeslices; 2) a graph view to reveal detailed content, such as the structure of G_i, to help users figure out why those particular patterns occur.

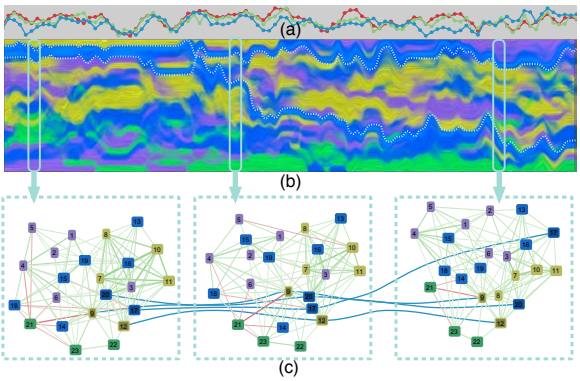

Figure 1: Overview: (a) three curves measuring the energy at each timeslice in the flow: e_v (red), e_c (blue), and e_r (green) (Sec. 4.4.1); (b) the flow view of a dynamic graph with two enhanced paths (Sec. 4.4.2); (c) the graph view showing three timeslices selected from the flow view with four nodes selected to show the splitting pattern (Sec. 5).

In the flow view (Fig. 1(b)), the x-axis represents time. Each individual timeslice G_i is summarized as "a colored bar" along the y-axis. In the adopted visual metaphor, colors play a very important role. The changes in color from left to right represent the changes of node activeness (measured by a graph metric) over time. With this feature, users can quickly get an overview of graph evolution patterns, such as where the graph changes smoothly or dramatically, even if they have no prior knowledge of the dataset. Generally, users are interested in timeslices that exhibit important changes. Inspired by the seam carving technique [2], we introduce an energy-based technique to measure the flow changes over time (Fig. 1(a)). The carving technique can be leveraged to emphasize critical timeslices by giving them more visual space. In addition, it can also be used to scale the flow visualization with a large number of time points by shrinking the flow in locations that exhibit fewer changes.

Once a user identifies important time points, he can select them from the flow view, and the related node-link diagrams are then displayed for comparison. In particular, we enhance them by visually linking the same nodes across different graphs (Fig. 1(c)).

4 Flow Visualization

In this section, we first introduce our flow design and considerations. Then we present the technical details, including vector field generation, vector field rendering, and interactions.

4.1 Design Considerations

Research on social network analysis has shown that graph metrics such as degree, betweenness, closeness, and cluster coefficient, are often used to summarize the structural features in a dynamic graph [33]. For example, node degree is an important metric of node activeness. Showing the changes in node degree at different levels can help users analyze the temporal behaviors of a single node (e.g., with a dramatic increase of activeness during a time period) or a group of nodes (e.g., becoming more inter-connected over time).

Furthermore, structural changes in a dynamic graph are measured not only by the value changes of graph metrics, but also by their rank changes over time. For example, in a social network, when a user's degree decreases, it may seem to indicate that s/he has become peripheral and dispensable. However, the user may have actually become more central since other users have lost their degrees more. It is therefore important to involve the rank factor into the visualization design to better illustrate the evolving patterns.

An intuitive way to visualize the metric changes is to leverage line charts. Fig. 2 shows an example of a line chart of a dynamic graph with 23 nodes and 165 timeslices. In the example, a line represents a node. At each time, the lines are sorted vertically based on their node degrees in that timeslice. As shown in Fig. 2, even for a small dynamic graph with only 23 nodes, the line-chart-based visualization is very cluttered due to there being so many line crossings. A neuroscientist who worked with us also confirmed that the cluttered line chart did not help her analysis (Sec. 6.1.2). Therefore, a new method was required to reduce visual clutter and allow users to freely control the levels of abstraction. Inspired by the distant resemblance between Fig. 2 and vector fields, we adopted the flow visualization technique to represent temporal metric changes. Previous research has shown that flow visualization is especially useful for gaining insight into very large, time-variant flow fields [26]. More importantly, flow visualization can naturally generate patterns at different abstraction levels by setting different parameters. We chose Line Integral Convolution (LIC) [13] as our flow visualization method because it provides not only a coherent global view of the data but also subtle details within. LIC also avoids clutter typically found in other types of vector field visualizations. For better performance, we use fast LIC [32] in *GraphFlow*.

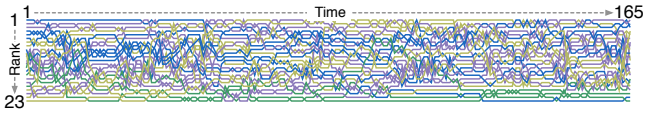

Figure 2: A line chart representation of a dynamic graph consisting of 23 nodes and 165 time points (the data of the youngest subject in Fig. 9). Nodes are represented by horizontal lines, which are sorted vertically based on the node degree at each time.

4.2 Vector Field Generation

We begin with a simple example consisting of two timeslices, G_1 and G_2 (Fig. 3), both of which contain the same vertices with their colors indicating a certain attribute value, such as category information.

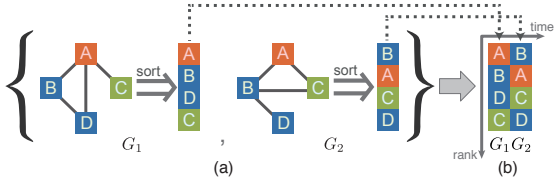

Figure 3: An example of turning a dynamic graph into a tile matrix: vertices at each timeslice are sorted by their degree values. Color is used to encode the category of each vertex.

To transform $\{G_1, G_2\}$ into a vector field, we first sort the vertices in each of them based on a graph metric, such as degree, betweenness, or the cluster coefficient metric. In Fig. 3(a), degree is used as an example. After every timeslice is processed, all the sorted vertex arrays are aligned from left to right based on their temporal order on a 2D plane (Fig. 3(b)). This creates colorful mosaic tiles. The next step is to create a vector for each tile. For each vertex in each timeslice, there is a tile whose vertical location represents its rank. When it does not exist in a timeslice, we consider it ranked as the smallest in the array. Thus, one intuitive way to generate vectors is to connect all the tiles corresponding to the same vertex in temporal order (Fig. 4(a)). The overall vector pattern can help users track the rank changes and understand the dynamics of the graph with respect to the metric (Fig. 4(b) and Fig. 4(c)).

Based on the description above, only one tile belongs to one vertex. Thus, there are two vectors in each tile: the incoming and outgoing vectors indicating the rank change for the corresponding vertex (Fig. 5(a)). To assign a vector to each pixel, we first divide each tile into four sub-tiles (Fig. 5(b)), each containing one vector (the incoming vector for the left sub-tiles, and the outgoing vector for the right sub-tiles). Then the vector at each pixel is calculated using an interpolation of their neighbors (Fig. 5(c) and Fig. 5(d)). Before choosing the number four for dividing the tiles, we tried several other choices, such as nine or sixteen. However, the results did not seem much different from the original, due to the multiple intermediate smoothing steps. Therefore, we chose the smallest number to produce the square sub-tiles.

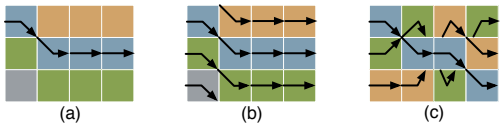

Figure 4: Vector generation and possible patterns: (a) vectors generated by connecting all the tiles corresponding to the same vertex in temporal order; (b) a dynamic graph changing smoothly; (c) a dynamic graph changing more dramatically.

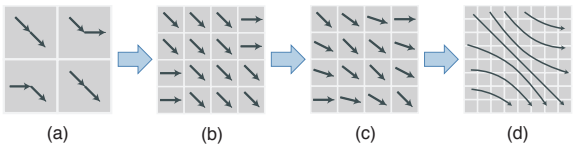

Figure 5: Assigning a vector to each pixel.

4.3 Vector Field Rendering

Our system employs fast LIC [32], an efficient texture-based flow visualization technique, to render the vector field as a continuous flow. By comparing the results before and after applying the method (Fig. 6(a) and (b)), we found that the result is improved by replacing the mosaic with smooth curves. However, directly applying LIC has two limitations in our application: 1) the main stream patterns can easily be interrupted by local patterns, such as a sudden fall/rise of a single vertex (see Fig. 6**A** for an example); and 2) colors of different trends can be readily mixed/convolved together, leading to false trend patterns (Fig. 6**B** for an example). To further improve the flow representation, two techniques are presented: 1) enhancing the main trends, and 2) differentiating the colors. Fig. 7 shows the pipeline of our rendering process.

4.3.1 Enhancing Main Trend

The basic idea is to group vectors of similar colors/directions and emphasize such groups by giving them the rendering priority necessary to make their overall patterns easily recognizable (**A** in Fig. 7).

We first introduce some preliminary definitions that are useful for subsequent discussions. Assuming we have an $n \times m$ matrix of tiles $\mathbf{T}$, each tile, denoted by $t_{i,j}$ ($1 \le i \le m$ and $1 \le j \le n$), contains:

- $\mathbf{v}_{i,j}$: the vector contained in $t_{i,j}$.
- $v_{i,j}$: the vertex in the dynamic graph, which is shared by multiple tiles in different timeslices.
- $c_{i,j}$: the color of $t_{i,j}$, indicating the category of $v_{i,j}$.
- $I_{i,j}$: the user-specified influence factor defining how much influence $\mathbf{v}_{i,j}$ has on its neighbors. The bigger the factor is, the more it impacts its neighboring vectors.
- $B_{i,j}$: the user-specified persistence factor determining how much $\mathbf{v}_{i,j}$ retains its own value under the influences of its neighbors.

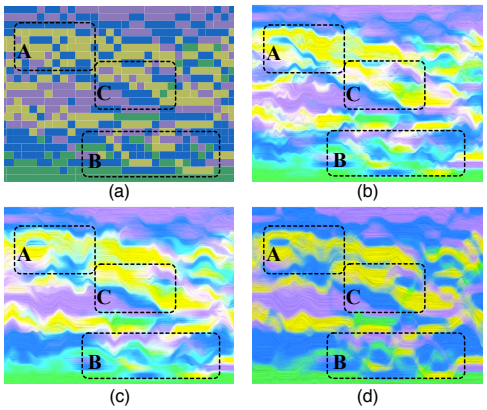

Figure 6: Comparison: (a) input tile matrix; (b) LIC result; (c) LIC result with enhanced main trends; (d) our final result (main trend enhancement + color differentiation + alpha blending).

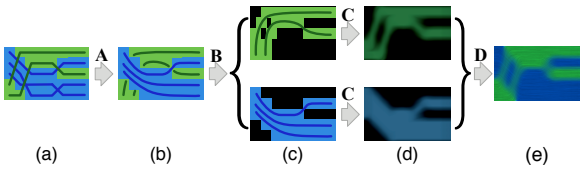

Figure 7: Rendering pipeline: (a) input tile matrix; (b) enhanced tile matrix; (c) refined tile matrix for individual colors; (d) LIC images for individual colors; (e) final result with alpha blending of all LIC images.

We use $\mathbf{P}_{i,j}$ to denote all tiles having the same vertex $v_{i,j}$, i.e., $\mathbf{P}_{i,j} = \{t_{x,y} | v_{x,y} = v_{i,j}, 1 \le y \le n, 1 \le x \le m\}$. The row index y of tile $t_{x,y}$ that belongs to $\mathbf{P}_{i,j}$ is denoted by $\mathbf{P}_{i,j}(x)$, i.e., $y = \mathbf{P}_{i,j}(x)$.

For each tile $t_{i,j}$, we use a consistency score $C_{i,j}$ to measure the similarity between $\mathbf{P}_{i,j}$ and its neighboring paths in the local area of $t_{i,j}$. The bigger $C_{i,j}$ is, the more $\mathbf{P}_{i,j}$ complies with the main trend at $t_{i,j}$. It is calculated as:

$$C_{i,j} = \frac{1}{R} \sum_{y=j-R_r}^{j+R_r} \left(\max_k \sum_{x=i-R_c}^{i+R_c} \frac{1}{1 + |\mathbf{P}_{i,y}(x) - \mathbf{P}_{i,j}(x) + k|} \right), \quad (1)$$

where R_c and R_r are the user specified factors that control the localness of the consistency score. The larger the factors are, the more global the feature that $C_{i,j}$ indicates. $R = (2R_r + 1)(2R_c + 1)$ is the normalizing factor that ensures $0 \le C_{i,j} \le 1$. Based on the consistency score, the new vector $\mathbf{v}'_{i,j}$ of tile $t_{i,j}$ can be defined as a weighted average of its neighboring vectors:

$$\mathbf{v}^*_{i,j} = \frac{B_{i,j}\mathbf{v}_{i,j} + \sum_{y=j-R_r}^{j+R_r} (I_{i,y}C_{i,y}|y-j|^{-1}\mathbf{v}_{i,y})}{B_{i,j} + \sum_{y=j-R_r}^{j+R_r} (I_{i,y}C_{i,y}|y-j|^{-1})}, \quad (2)$$

If the vectors of a local area have similar directions, they should be assimilated and enhanced as a group. Based on the dataset quality, the calculation could be done many times iteratively to obtain a smooth vector field.

Another part of the enhancement is to remove small, isolated color blocks that interrupt the main trend patterns (see Fig. 6**C** for an example). Similar to the vector smoothing process, the new color of $t_{i,j}$ is given by:

$$c^*_{i,j} = \operatorname*{argmax}_c \frac{B_{i,j}\mathbf{1}_{\{c_{i,j}\}}(c) + \sum_{y=j-R_r}^{j+R_r} \frac{(I_{i,y}C_{i,y}\mathbf{1}_{\{c_{i,y}\}}(c))}{|y-j|}}{B_{i,j} + \sum_{y=j-R_r}^{j+R_r} \frac{(I_{i,y}C_{i,y})}{|y-j|}}, \quad (3)$$

where $\mathbf{1}_{\{c\}}(x)$ is the indicator function defined as:

$$\mathbf{1}_A(x) = \begin{cases} 1 & \text{if } x \in A, \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

4.3.2 Differentiating Colors

Another limitation of using LIC in our application is that it only allows one vector for each pixel. Therefore, trends with different directions cannot co-exist at the same location. For example, Fig. 6**B** shows three colored trends that cross: green, yellow, and blue. Simply applying LIC to it would convolve all three colors together, leading to the wrong flow trends (Fig. 6(b) and Fig. 6(c)).

We have solved this problem by running LIC in multiple layers for individual colors. For each color c, a new tile matrix $\mathbf{T}^c$ is derived from the enhanced tile matrix, denoted by $\mathbf{T}^*$. For each tile with color c, we search backwards and forwards along the direction of its vector v to fill more empty tiles with the same color c and vector v (step **B** in Fig. 7). Then LIC is applied on $\mathbf{T}^c$ to get a flow image for only color c (step **C** in Fig. 7).

We merge all layers using alpha blending to get the final result (step **D** in Fig. 7). From this result, we can clearly see that the false flows are not only eliminated, but the remaining flows become clearer and more fluent (Fig. 6(d)**B**).

Although our color rendering technique can help users differentiate flow trends of individual colors, it still suffers from some constraints when using color to represent categorical information. For example, there are a limited number of colors that can be used to effectively represent different categories [22, 36]. Our alpha blending method will reduce the number further. In our experiments, we found that users generally do well using five or six colors in our system. On the other hand, different colors may create different visual impacts, which may cause misleading information, such as false emphasis. To alleviate this problem, besides providing two standard color encoding schemes, we also allow users to manually assign different colors to the categories they are interested in.

4.4 Interaction

GraphFlow allows users to interact with the flow view and to examine relevant data from multiple perspectives. Specifically, we have designed two interactions: flow carving and path enhancing.

4.4.1 Flow Carving

We designed flow carving, inspired by seam carving [2], to quantitatively measure the change degree of each timeslice. It serves two purposes. First, it helps users easily find critical timeslices, such as those with a dramatic change. Second, it can be used to remove vertical slices with smooth changes, so that screen space can be better allocated to more important information.

Mathematically, we define the energy for a timeslice $e(G_i)$ as $\sum_{j=1}^{n} e(t_{i,j})$, where $e(t_{i,j})$ is the energy function that measures the energy of the local change at tile $t_{i,j}$. We have examined several possible rank change measures, including the gradient, saliency measure [2], entropy [14], and the inversion number [25].

The gradient, saliency measure tries to capture the vector change at a local area of $t_{i,j}$ [2]:

$$e_v(t_{i,j}) = \sum_{x=i-k}^{i+k} \sum_{y=j-k}^{j+k} ||\mathbf{v}_{i,j} - \mathbf{v}_{x,y}||, \quad (5)$$

Entropy can measure the color consistency around $t_{i,j}$ [14]:

$$e_c(t_{i,j}) = -\sum_c p^k_{i,j}(c) \log p^k_{i,j}(c), \quad (6)$$

where $p^k_{i,j}(c) = k^{-2} \sum_{x=i-k}^{i+k} \sum_{y=j-k}^{j+k} \mathbf{1}_{\{c\}}(c_{x,y})$.

The inversion number is used to measure the sortedness of a sequence [25]. Formally, the inversion number for tile $t_{i,j}$ is defined as:

$$e_r(t_{i,j}) = \sum_{y=1}^{n} \mathbf{1}_{\{-1\}}\big(\operatorname{sgn}\big((\mathbf{P}_{i,y}(i-1) - \mathbf{P}_{i,j}(i-1)) \times (y-j)\big)\big), \quad (7)$$

where $\mathbf{1}_{\{-1\}}(x)$ is an indicator function defined in Eq. 4. Fig. 1(a) compares the results of these three energy functions (e_v as red, e_c

as green, and e_r as blue) on the same flow. As expected, no single measure works well across all times, but they have similar temporal behaviors. According to the experiments, we found the gradient, saliency measure works better in most cases. Thus, we have adopted this measure in *GraphFlow*.

4.4.2 Path Enhancing

Although the flow representation provides a nice overview of the metric changes, it is ineffective at identifying individual paths. To tackle this issue, a path enhancing technique was designed.

The basic idea is to enhance the persistence and influence factors of each tile of the path of interest. As described in Sec. 4.3.1, $B_{x,y}$ and $I_{x,y}$ represent the persistence and influence factors for tile $t_{x,y}$. By increasing them, $t_{x,y}$ is more likely to retain its original color and vector and to have more influence on its neighbors.

For example, when a user finds an interesting region in the flow representation, and wants to know where this path comes from or goes to, s/he can simply click it. Our system then retrieves the related tiles $t_{i,j}$ and path $\mathbf{P}_{i,j}$. For each tile $t_{x,y}$ that belongs to $\mathbf{P}_{i,j}$, we increase $B_{x,y}$ and $I_{x,y}$, and re-generate the flow image. Fig. 1(b) shows two examples of path enhancement (marked as dotted lines).

4.4.3 Details-on-Demand

Once users identify an interesting pattern in the flow view, they can click on the point of interest. Our system will locate the corresponding tile and reveal more information to users, such as the graph structure at that time point or related details that may be different from application to application.

5 Graph Layout

When a user selects several timeslices from the flow visualization, the corresponding graphs are presented side by side. Then the user can examine and find the particular reasons why those timeslices are so interesting. To achieve this, two techniques have been developed to help users easily track the nodes of interest across multiple graphs [11, 24].

One widely adopted approach is to lay out each graph by keeping the relative positions of unchanged vertices/edges as stable as possible. In our system, we use the energy minimizing method introduced in [5] to generate the dynamic layouts. For a timeslice $G_t = (V_t, E_t)$, the energy function is formulated as:

$$S_t = \sum_{(v_i,v_j)\in E_t} \omega_{ij}(d_{ij,t} - ||\mathbf{x}_{i,t} - \mathbf{x}_{j,t}||^2) + \omega||\mathbf{x}_{i,t} - \mathbf{x}_{i,t-1}||^2, \quad (8)$$

where $\omega_{i,j}$ and ω are the user-specified weight factors. $d_{ij,t}$, and $\mathbf{x}_{i,t}$ are the ideal distance for edge (v_i, v_j) and the position of vertex v_i at timeslice G_t, respectively.

In addition, we draw curves to visually connect the same vertices across different graphs (Fig. 8) and bundle the links together to provide visual aids and help users discover patterns within a group of vertices. In our system, the bundling result is achieved by turning the straight links into curves based on a force model introduced in [19]. Fig. 8 shows an example in which two links connect two graphs: G_i and G_{i+1}. For each link connecting the same vertex in G_i and G_{i+1}, we put two subdivision points on the link in the middle. These points are aligned vertically based on which graph they are close to. A linear, attracting spring force $\mathbf{F}_s$ is used between the shifted subdivision point and its original position. On the other hand, an attracting force $\mathbf{F}_e$ is used between each pair of subdivision points that are on the same vertical line. Therefore, the total force exerted on p_i is calculated as:

$$\mathbf{F}_{p_i} = k(\mathbf{p}_i^o - \mathbf{p}_i) + \sum_{\mathbf{q}\in P/\{\mathbf{p}_i\}} \frac{(\mathbf{q}-\mathbf{p}_i)}{||\mathbf{q}-\mathbf{p}_i||^2}, \quad (9)$$

where $\mathbf{p}_i$ and $\mathbf{p}_i^o$ are the actual and original locations of p_i, respectively. P contains the locations of all the subdivision points on the same vertical line as p_i.

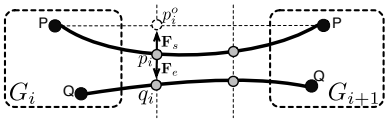

Figure 8: An example of force settings for two links.

6 Case Study

Two different datasets were adopted to demonstrate the versatility of *GraphFlow*. Despite their differences in structure and content, *GraphFlow* can easily be applied to both, unveiling interesting patterns that we describe in our two case studies.

The first case study, conducted on the functional brain connectivity dataset from an MRI scanner, demonstrated how domain experts can quickly discover patterns in their data using *GraphFlow*. The second case study, covering two breaking events in Twitter, describes how the flow view uses different metrics to help users understand the overall evolving patterns in each event and locate critical time points during these events.

All the flow images in this paper took up to 0.6 seconds to generate on a desktop computer with an Intel Quad-Core 2.80 GHz CPU and 8G RAM memory, based on image quality and data size.

6.1 Functional Brain Connectivity Dataset

This case study was conducted with a team of neuroscientists at the University of Washington. We performed the case study over a couple of months, meeting with the team about twice a month and improving the prototype after each visit. Our main contact was Mary, who is a neuroscientist investigating brain aging and attempting to characterize the effect of aging on cognitive abilities.

6.1.1 Data and Tasks

Higher cognitive abilities (memory, reasoning ability, etc.) are neither the result of activity strictly localized in specific neural structures, nor of the brain as a whole. They emerge from the coordination of distributed networks (groups of neurons) of cortical regions. When a subject is resting in a scanner, their blood-oxygenated level-dependent signal measured by functional magnetic resonance imaging (fMRI) shows regional patterns of correlations. These patterns recreate maps of known, large-scale brain networks that are activated when the subject performs tasks in the scanner. Because of the similarity, the correlation strength between regions is thought to be related to the efficiency of communication between corresponding regions.

The team analyzed fMRI data using R. Team members extracted mean timecourses from cortical regions of interest (ROIs), defined sliding windows, and computed correlations between ROIs within each sliding window. In addition to plotting various means and variances in R, the visualization they created for studying the functional connectivity graph was a dynamic matrix of correlations. The particular subset of data that they focused on comprised 23 ROIs, categorized into four groups. These groups represent four different large-scale brain networks, respectively. Three of them are involved in the dorsal attentional network, the fronto-parietal task control network, and the salience network, while the fourth is active when not engaged in a task (default mode network). Fig. 11 shows one example of graph construction. 23 nodes represent 23 ROIs with their background color representing their group attribute. At a time point, if two nodes (i.e., ROIs) are correlated, they have an edge connecting them.

Despite such a small graph size, neuroscientists reported that finding patterns in the animated matrix was extremely difficult, even when pausing the animation and playing it multiple times. In particular, they commented that it was difficult for them to identify high-level patterns involving more than a couple of ROIs, such as

comparing the evolution of intra-connectivity (within a subset of ROIs) and inter-connectivity (between several groups of ROIs).

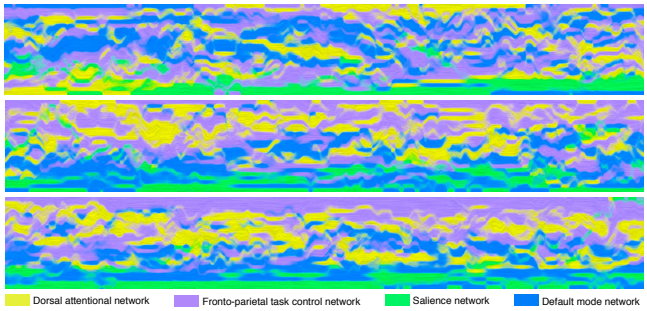

Figure 9: Functional brain connectivity summaries of three different subjects from younger to older (from top to bottom).

6.1.2 Insight discovery

We provided Mary with our prototype, iteratively improving it to better handle her data. Fig. 9 shows the very first dataset Mary loaded in *GraphFlow*. These flow diagrams represent a subset of the functional connectivity graph of three different subjects, ranging from younger to older. When studying these visualizations, the team immediately commented on the higher-level patterns of connectivity. In particular, they were excited about the pattern exhibited by the yellow flow, representing the dorsal attentional network. They observed that the overall degree of the brain regions in this network tended to decrease (relative to other networks) in older subjects. This finding seems to indicate that a pattern of lower connectivity among nodes in the dorsal attentional network is related to brain aging. While it would require extensive statistical validation and multiple observations to validate this hypothesis, *GraphFlow* could successfully lead to such an insight shortly after loading the data into the tool.

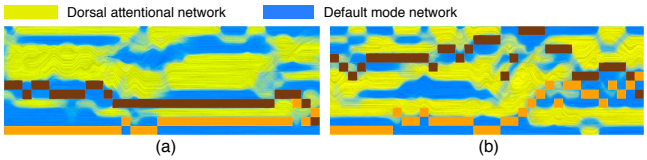

Figure 10: The degree changes of two nodes over time (highlighted as brown and orange) in two subjects: (a) similar evolution patterns in the middle-aged subject; (b)dissimilar trends in the older subject.

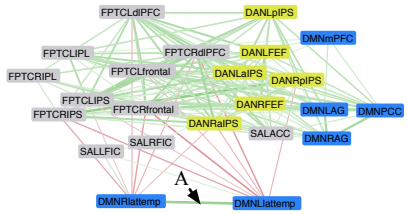

Figure 11: The node-link diagram shows the behavior of two nodes of the default mode networks. While these two nodes are strongly correlated to each other (green links), they are also negatively correlated to the nodes of the other two regions (the gray regions).

Fig. 10 illustrates correlations between nodes in two well-characterized networks, the default mode network (colored blue) and the dorsal attentional network (colored yellow), obtained while a subject was at rest. The evolution patterns of a middle-aged subject are shown in Fig. 10(a) and those of an older subject in Fig. 10(b). Mary commented that the flow diagrams illustrated a broad pattern of dynamic connectivity, expressed in terms of the varying degree of each node in a network over time. She observed that Fig. 10(a) shows that nodes in the dorsal attentional network (yellow) tend to be more connected during this scan than those in the resting state network (blue). In contrast, in Fig. 10(b), the relative degrees of node connectivity within the default mode network and the dorsal attentional network varied significantly throughout the duration of the scan.

In addition, Mary commented on two particular nodes in the default mode network: the posterior cingulate cortex and the medial prefrontal cortex, which are highlighted as orange and brown in Fig. 10, respectively. She was intrigued to notice that in the middle-aged subject (Fig. 10(a)), they tended to have similar ranks in degree of connectivity over time whereas they tended to have dissimilar degrees of connectivity over time in the older subject (Fig. 10(b)). Intrigued by the behavior of these two neural regions, Mary further investigated their connections to the rest of the brain. Fig. 11 shows the new pattern she discovered in the middle-aged subject: these two nodes are strongly correlated with each other (green link marked as **A**) but negatively correlated (red links) to nodes in attentional networks (purple and orange nodes). This is consistent with vast literature describing the complimentary roles of the dorsal attentional network and the resting state network, which tend to be negatively correlated. However, the variability of this relationship during a single scan and between subjects is something that *GraphFlow* helped them discover at a higher level than previous tools allowed.

In a very short time, they could identify interesting patterns they did not know about before that could possibly lead to important discoveries on the nature of the brain and the effects of aging. Mary commented that she greatly favored the processed flow diagrams (Fig. 6(d)) over the flow charts (Fig. 4(c)) composed of vectorial lines (we include both options in the tool). The cluttered line chart distracted her from her analysis tasks. The team made multiple comments on the aesthetically pleasing look of *GraphFlow* and they were all very excited about the higher-level patterns such visualizations could exhibit. We are continuing the collaboration with the team and expect new discoveries in the near future.

6.2 Twitter Dataset

In this case study, we explored the evolving patterns of Twitter accounts during two social events. The first dataset, covering "the death of Osama bin Laden," contains 910,429 tweets spanning May 1st 10:20pm EST, 2011 to May 2nd 2:20am EST, 2011. The second, related to the basketball event known as "Linsanity," contains 1,305,906 tweets spanning Feb. 1st 12:00am EST, 2012 to Feb. 14th 12:00am EST, 2012.

The dynamic graph for each dataset is defined as follows: if two different accounts exist in one single tweet (either content or screen name), they have one edge at the posting time of the tweet. The time steps for the two datasets are one minute and one day, respectively. To simplify the exploration, we collected and categorized the top 100 accounts having the most edges during the time period for each dataset (Table 1).

	bin Laden Data	Linsanity Data
News media (green)	31	31
Journalist (blue)	17	16
Celebrity (orange)	39	38
Imposter (purple)	3	0
Others (yellow)	10	15

Table 1: Number of Twitter accounts in the top 100 under each category for the bin Laden dataset and Linsanity dataset.

Fig. 12(a) and Fig. 12(c) show the flow summaries of both datasets, in which vertices are sorted by degree. Intuitively, the higher a vertex is in the flow image, the more active it is during the event, in terms of posting tweets or being mentioned by other tweets.

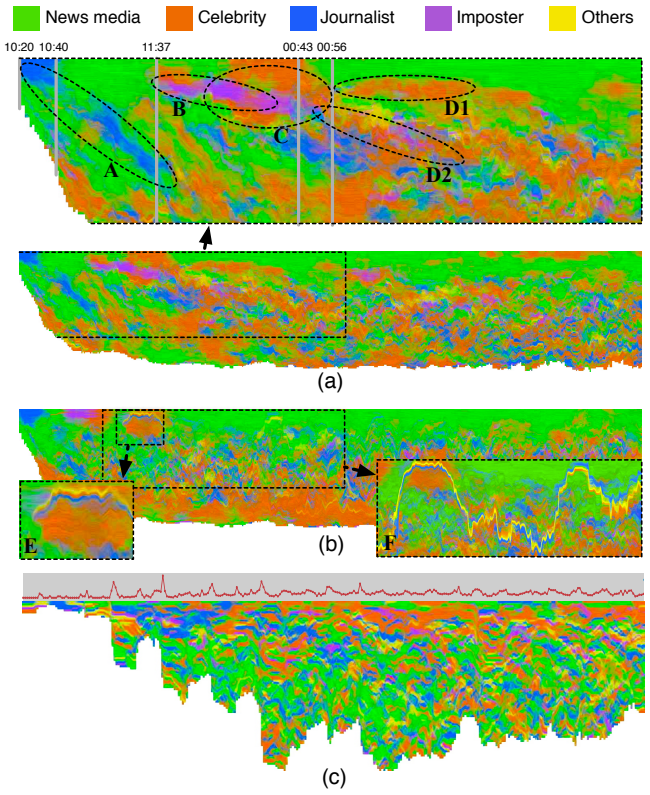

Figure 12: Flow summaries of the twitter datasets: (a) summary of bin Laden data (sorted by degree); (b) summary of bin Laden data (sorted by closeness); (c) summary of Linsanity data (sorted by degree).

As shown in Fig. 12(a), the overall pattern becomes more and more complex as time goes on. In the first few minutes (from 10:20pm to 10:40pm), journalist accounts (blue), such as @keithurbahn, @brianstelter, and @jacksonjk, were dominant. This is because they posted rumors of Osama Bin Laden's death, such as *"I'm told by a reputable person they have killed Osama Bin Laden,"* which were widely retweeted as people were wondering about their reliability. But very soon (around 10:40pm), media accounts (green), such as @cnnbrk, @nytimes, and @cbsnews, confirmed the rumors, with tweets such as *"NYT NEWS ALERT: Osama bin Laden Is Dead, White House Says"* by @nytimes. They immediately took over the leading positions, because people stopped retweeting the rumors and started retweeting information from the reliable sources. Thus, the journalist accounts (blue) declined in the flow representation (marked as **A** in Fig. 12(a)).

Once the rumors had been confirmed and spread throughout the Internet, people began to change their focus. One interesting pattern is related to three imposters (purple), @real_bin_laden, @osamabinladen, and @osamabinladen96, which we found among the top 100 accounts. They were very active for a short period (from 11:37pm to 00:43am) after the rumors were confirmed (marked as **B** in Fig. 12(a)). By extracting their posting history, we found that they actually posted messages as early as 10:38pm such as *"Relax, everyone, I'm just faking my own death..."* But they only became popular after the rumors were confirmed. This may be partially due to people being more curious about the reliability of the rumors at the beginning, and paying no attention to those imposters. Once the rumors were confirmed, people were digging everywhere and found those imposters funny and worth mentioning; however, their jokes quickly wore out and fell out of the spotlight.

The third wave (marked as **C** in Fig. 12(a)) of leading accounts comes from celebrities (orange). We extracted some tweets there and found that they are mostly joking about bin Laden, such as *"R.I.P to the king of hide-n-seek Osama Bin Laden."* In particular, we found a very clear splitting pattern around 00:56am for the celebrity accounts (marked as **D1** and **D2** in Fig. 12(a)). After examining the tweets from both branches, we found **D1** to be very focused, involving similar tweets, such as *"BREAKING NEWS: Donald Trump demands Osama Bin Laden's death certificate."* On the other hand, **D2** is from miscellaneous tweets. So we were curious why **D1** is split from the rest. After checking the content in **D1** and searching the Web, we found that **D1** is related to a political joke caused by Obama's official TV announcement interrupting the TV show "Celebrity Apprentice" hosted by Donald Trump, who had challenged Obama about his birth certificate at that time. People found such a coincidence funny, so they made several jokes about Trump, Obama, and the death certificate.

In addition to showing the degree metric, we also generated a flow image using the closeness centrality metric (Fig. 12(b)). It is clear that Fig. 12(a) and Fig. 12(b) have a certain similarity at the top. In addition to the similarity, we also found some interesting outlier paths in Fig. 12(b). For example, we found an outlier (Fig. 12(b)**E**) pattern buried in a group of celebrity accounts. It includes three highly correlated accounts, i.e., @darrenrovell, @jtalarico328, and @dvnjr, from three different categories (blue, orange, and yellow). This attracted our attention because the three accounts co-occurred with each other for a period of time. So we enhanced all three accounts and it turns out they co-occurred almost everywhere in the flow (Fig. 12(b)**F**).

By retrieving the related content, we found that all three accounts stayed connected because they all appeared in a single tweet *"RT @darrenrovell: May 1, 1945: Hitler confirmed dead. May 1, 2011: Bin Laden confirmed dead. (via @JTalarico328, @DVNJr)."* To further investigate the underlying pattern in this tweet, we extracted the order of retweeting, and found the tweet was first created by @JTalarico328, retweeted by @dvnjr 12 minuteslater, and finally by @darrenrovell after another 3 minutes. This example clearly shows how interesting information is propagated: from an ordinary person to a journalist to a very visible celebrity and finally to the general public.

Compared with the bin Laden data, the Linsanity data shows a different pattern (Fig. 12(c)). In this event, the news media (green) was not dominant. Instead, celebrity accounts (orange) were a relatively stable dominant group. We then calculated the flow carving curve based on the inversion number. Several peaks occurred at the beginning. After examining the time points, we found high correlations between those peaks and Jeremy Lin's game schedule. In particular, we noticed that the whole flow was generally triggered by the first game (Feb. 4th 07:30pm), and after several peaks in activity, "Linsanity" became a normal topic, the curve became smoother, and the game schedule contributed less to the curve fluctuations.

7 Discussion and Future Work

In this paper, we addressed the problem of exploring a dynamic graph with a static method. Accordingly, we introduced a novel flow-based visualization design for summarizing high-level evolution patterns in a dynamic graph. The key idea is to convey changes in the structure of a graph through the evolution of a number of graph metrics computed on its nodes/edges. Although we used degree and closeness as examples throughout this paper, *GraphFlow* does not depend on any particular graph metrics. Other metrics, such as cluster coefficient and triangle number, can also be directly applied in our system. To further aid users in information seeking, a trend analysis was designed to identify the critical time points in the flow visualization. Furthermore, detailed structural changes across several correlated time points were provided to examine the major causes that lead to such interesting patterns. Two case studies were conducted to demonstrate the usefulness and effectiveness of our system.

Our flow design does have some limitations. First of all, the patterns highly depend on the metric adopted. In our case studies,

we used common metrics. However, in some special scenarios, common metrics may not be meaningful or adequate. Choosing the most appropriate metrics will highly depend on a user's domain knowledge. Second, in certain applications where the absolute values are critical, our flow representation, which mainly focuses on rank changes, may not be very helpful. To address this issue, a standard line chart can be combined with our flow representation to show absolute values for selected nodes. Third, our graph view can only work well for small graphs, which limites the scalability of the whole system. To support large graphs, LOD-based [38] or DOI-based approaches [34] can be leveraged.

In the future, we plan to combine some of the graph metrics for more complex graphs, such as weighted graphs. In addition to flow carving techniques that address the scalability issue on the time dimension, we may also investigate techniques for the scalability issue on graph size, such as aggregating nodes based on the clustering or hierarchical structure of the nodes. On the other hand, it is not conventional to use the flow visualization to visualize a dynamic graph. In the future, we plan to design a series of controlled experiments to systematically evaluate how people accept, consume, and interpret such unfamiliar visual representations.

REFERENCES

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

[2] S. Avidan and A. Shamir. Seam carving for content-aware image resizing. *ACM Transaction on Graphics*, 26(3):1–9, 2007.

[3] G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph drawing: Algorithms for the visualization of graphs*. Prentice-Hall, 1999.

[4] F. Beck, M. Burch, C. Vehlow, S. Diehl, and D. Weiskopf. Rapid serial visual presentation in dynamic graph visualization. In *IEEE Symposium on Visual Languages and Human-Centric Computing (VLHCC)*, pages 185–192, 2012.

[5] K. Boitmanis, U. Brandes, and C. Pich. Visualizing internet evolution on the autonomous systems level. In *Graph Drawing*, pages 365–376, 2007.

[6] U. Brandes and S. R. Corman. Visual unrolling of network evolution and the analysis of dynamic discourse. In *IEEE Symposium on Information Visualization, 2002*, pages 145–151, 2002.

[7] U. Brandes and B. Nick. Asymmetric relations in longitudinal social networks. *IEEE Transaction on Visualization and Computer Graphics*, 17(12):2283–2290, 2011.

[8] U. Brandes and D. Wagner. Tracking multiple independent targets: Evidence for a parallel tracking mechanism. *Spatial Vision*, 3(3):179–197, 1988.

[9] J. Branke. Dynamic graph drawing. In *Drawing Graphs*, pages 228–246, 1999.

[10] M. Burch, C. Müller, G. Reina, H. Schmauder, M. Greis, and D. Weiskopf. Visualizing dynamic call graphs. In *Proceedings of Workshop on Vision, Modeling, and Visualization*, pages 207–214. The Eurographics Association, 2012.

[11] M. Burch, C. Vehlow, F. Beck, S. Diehl, and D. Weiskopf. Parallel edge splatting for scalable dynamic graph visualization. *IEEE Transaction on Visualization and Computer Graphics*, 17(12):2344–2353, 2011.

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

[13] B. Cabral and L. C. Leedom. Imaging vector fields using line integral convolution. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '93, pages 263–270, New York, NY, USA, 1993. ACM.

[14] T. M. Cover and J. A. Thomas. *Elements of information theory*. Wiley-interscience, 2006.

[15] C. Erten, P. J. Harding, S. G. Kobourov, K. Wampler, and G. V. Yee. Graphael: Graph animations with evolving layouts. In *Graph Drawing*, pages 98–110, 2003.

[16] C. Friedrich and M. E. Houle. Graph drawing in motion ii. In *Graph Drawing*, pages 220–231, 2001.

[17] N. Henry and J.-D. Fekete. Matrixexplorer: A dual-representation system to explore social networks. *IEEE Transaction on Visualization and Computer Graphics*, 12(5):677–684, 2006.

[18] N. Henry, J.-D. Fekete, and M. J. McGuffin. Nodetrix: A hybrid visualization of social networks. *IEEE Transaction on Visualization and Computer Graphics*, 13(6):1302–1309, 2007.

[19] D. Holten and J. J. van Wijk. Force-directed edge bundling for graph visualization. *Computer Graphics Forum*, 28(3):983–990, 2009.

[20] S. Kairam, D. MacLean, M. Savva, and J. Heer. Graphprism: Compact visualization of network structure. In *Advanced Visual Interfaces*, pages 498–506, 2012.

[21] G. Kumar and M. Garland. Visual exploration of complex time-varying graphs. *IEEE Transaction on Visualization and Computer Graphics*, 12(5):805–812, 2006.

[22] A. Light and P. J.Bartlein. The end of the rainbow? color schemes for improved data graphics. *EOS Transactions of the American Geophysical Union*, 85(40):385–391, 2004.

[23] G. Liu, E. Austen, K. Booth, B. Fisher, M. Rempel, and J. T. Enns. Multiple object tracking is based on scene, not retinal, coordinates. *Journal of Experimental Psychology: Human Perception and Performance*, 31(2):235–247, Apr. 2005.

[24] K. Misue, P. Eades, W. Lai, and K. Sugiyama. Layout adjustment and the mental map. *Journal of visual languages and Computing*, 6(2):183–210, 1995.

[25] P. Mutzel and J. Michael. Simple and Efficient Bilayer Cross Counting. *Journal of Graph Algorithms and Applications*, 8(2):179–194, 2004.

[26] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch. The state of the art in flow visualisation: Feature extraction and tracking. *Computer Graphics Forum*, 22(4):775–792, 2003.

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

[28] M. Rosvall and C. T. Bergstrom. Mapping change in large networks. *PloS one*, 5(1):e8694, 2010.

[29] A. Sallaberry, C. Muelder, and K.-L. Ma. Clustering, visualizing, and navigating for large dynamic graphs. In *Graph Drawing*, pages 487–498, 2012.

[30] C. Shi, W. Cui, S. Liu, P. Xu, W. Chen, and H. Qu. Rankexplorer: Visualization of ranking changes in large time series data. *IEEE Trans. Vis. Comput. Graph.*, 18(12):2669–2678, 2012.

[31] L. Shi, N. Cao, S. Liu, W. Qian, L. Tan, G. Wang, J. Sun, and C.-Y. Lin. Himap: Adaptive visualization of large-scale online social networks. In *PacificVis*, pages 41–48, 2009.

[32] D. Stalling and H.-C. Hege. Fast and resolution independent line integral convolution. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 249–256. ACM, 1995.

[33] R. Toivonen, L. Kovanen, M. Kivelä, J.-P. Onnela, J. Saramäki, and K. Kaski. A comparative study of social network models: Network evolution models and nodal attribute models. *Social Networks*, 31(4):240–254, 2009.

[34] F. Van Ham and A. Perer. Search, show context, expand on demand: Supporting large graph exploration with degree-of-interest. *IEEE Transaction on Visualization and Computer Graphics*, 15(6):953–960, 2009.

[35] F. van Ham, H.-J. Schulz, and J. M. DiMicco. Honeycomb: Visual analysis of large scale social networks. In *Human-Computer Interaction INTERACT 2009*, pages 429–442, 2009.

[36] C. Ware. *Information visualization: perception for design*. Morgan Kaufmann, 2012.

[37] M. Wattenberg. Visual exploration of multivariate graphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 811–819, 2006.

[38] M. Zinsmaier, U. Brandes, O. Deussen, and H. Strobelt. Interactive level-of-detail rendering of large graphs. *IEEE Transaction on Visualization and Computer Graphics*, 18(12):2486–2495, 2012.