Spatiotemporally Coherent Time-Varying Graph Drawing with Multi-Focus+Context

Kun-Chuan Feng, Chaoli Wang, *Member, IEEE*, Han-Wei Shen, and Tong-Yee Lee, *Member, IEEE*

***Abstract*—Graph drawing is a standard method to visualize relational information. Many previous approaches have been focused on the design of layout for static graphs to achieve good readability. Naïvely utilizing these approaches to layout the graphs in individual time steps of a time-varying data set, however, often fails to maintain spatiotemporal coherence, thus making it difficult for the viewers to track the evolution of salient features in the graph. This situation is exacerbated by the ever-growing data size. To address this issue, we propose a new approach for time-varying graph drawing that can achieve both spatiotemporal coherence and focus+context visualization. Our approach utilizes existing graph layout algorithms to produce the initial graph layout, and formulates the problem of generating spatiotemporally coherent time-varying graph visualization as a deformation optimization problem. The optimization is achieved by incorporating spatiotemporal coherence constraints and adopting the concept of super graphs to preserve spatiotemporally coherent content. The proposed deformation framework can balance the needs for aesthetic quality and dynamic stability when visualizing time-varying graphs with interactive performance. Our method is particularly useful for multi-focus+context visualization of time-varying graphs, and can prevent the graph nodes of focus from having abrupt changes in size and location in the time sequence. Experiments demonstrate that our method can maintain good spatiotemporal coherence and produce stable results, thus providing a more engaging viewing experience for the users.**

***Index Terms*—Graph drawing, time-varying graphs, spatiotemporal coherence, focus+context visualization.**

I. Introduction

EFFECTIVE drawing of graphs plays an increasingly important role in data understanding for many science and engineering disciplines such as biology, archaeology, information retrieval, and VLSI circuit design. More recently, graphs have also been applied to problems in various areas of social computing such as visualizing online social networks and analyzing terrorist networks and organizations. To date, most of the existing graph layout algorithms are primarily focused on static graphs. Examples include the force-directed graph layout [7], [23] and many of its variants such as GVA [11] and LinLog [29]. There also exist other efficient algorithms that use a multi-scale approach such as the work of Cohen [3], Hachul and Jünger [18], and GRIP [14]. Some of the faster graph layout algorithms are available in the form of algebraic layouts such as ACE [24] and the high-dimensional embedding algorithm [18].

Despite the advances in graph layout as described above, the more challenging issue of time-varying graph layout has not received full attention. Many graphs are dynamic in nature, such as event graphs extracted from an archive showing the connection and evolution of events over time, processor communication graphs obtained from a supercomputer run, and friendship networks inferred from a social website. To effectively examine hidden patterns and discover new insights into the data from a time-varying graph, a key component to consider when designing the graph layout is to maintain spatiotemporal coherence of graph nodes and edges across time steps so that their temporal evolution and correlation can be clearly revealed. The required spatiotemporal coherence properties include: (1) nodes keep their locations similar to the previous time step as much as possible, especially for nodes of high importance, (2) successive frames display smooth transitions when the graphs are evolving over time with edges and nodes being added or deleted, and (3) the layout of the graph should be of high aesthetic quality. Simply applying a static graph layout algorithm to graphs of individual time steps will not maintain sufficient spatiotemporal coherence, and hence, the resulting visualization will suffer from undesired artifacts such as flickering or popping (i.e., abrupt changes of nodes or edges in size or location). These artifacts make it difficult for viewers to track the changes of graph, thus hindering data understanding.

To further assist the understanding of complex time-varying graphs, besides maintaining spatiotemporal coherence, another critical component for effective visualization is to have the ability to perform focus+context viewing. Focus+context (F+C) visualization stems from the need to show both overview (context) and detailed (focus) information simultaneously within a limited display area. It is particularly important for visual analysis and interactive navigation of a large-scale time-varying graph. With the aid of focus+context techniques, tracking individual nodes of interest and inferring the changes of relations can be done much more easily. Although previously solutions have been proposed for F+C visualization of static graphs [15], [31] or static data such as polygons [39] or volume data [40], the problem of spatiotemporally coherent F+C visualization of time-varying graph has not been fully investigated.

In this paper, we propose a novel approach for time-varying graph drawing that offers a more engaging viewing experience for users through F+C visualization with spatiotemporal coherence. Specifically, we formulate the time-varying graph

K.-C. Feng and T.-Y. Lee are with National Cheng Kung University, Tainan, Taiwan, ROC. Email: stevenf3@gmail.com; tonylee@mail.ncku.edu.tw.

C. Wang is with Michigan Technological University, Houghton, MI 49931. Email: chaoliw@mtu.edu.

H.-W. Shen is with The Ohio State University, Columbus, OH 43210. Email: hwshen@cse.ohio-state.edu.

layout problem as a deformation optimization problem with an initial layout generated from other graph layout algorithms. To generate the desired time-varying layout with F+C visualization, we incorporate the concept of super graph [6] and solve a series of spatiotemporal coherence constraints to preserve spatiotemporally coherent contents. Our method allows the users to specify multiple foci in their visualization. Our solution can produce smooth F+C visualization by preventing the graph nodes in the foci from having abrupt changes in size and location over time while keeping the context information as stable as possible. We demonstrate the efficacy of our method with experimental results to validate the success of our approach.

II. Related Work

Graph layout is an important topic in information visualization and has been an active research area for many years. Survey of graph layout methods and performance comparisons can be found in [4], [19], [22].

Closely related to our work are those on time-varying graph drawing and F+C graph visualization. In addition, static graph layout algorithms are also related as solving dynamic graph drawing can often be reduced to static graph drawing.

A. Layout Algorithms for Static Graphs

Visual Pattern-Based Layout. The goal for the algorithms of this type is to generate layouts with familiar visual patterns so that the viewers can quickly understand the graph structure. Simple examples include the circular layout and radial layout. More complicated examples include the Sugiyama layout [34] which places nodes on a few parallel lines, each of which represents a "layer". Munzner [28] proposed to extract a spanning tree from a graph as a simplified structure, and to layout the nodes in each tree level with a sphere/circle packing pattern. Recently, treemaps and space-filling curves have also been used as the visual patterns to layout a graph [26], [27].

Energy-Based Layout. A classical example of energy-based layout is the force-directed layout. Since the force-directed layout algorithm was introduce by Eades in [7], several variants were proposed. Among them, the most well known methods are the KK-layout [23] and the FR-layout [11]. In recent years, more complex models were proposed, including LinLog (a clustering energy model) [29], high-dimensional embedding [21], and stress majorization [16].

Trees & Graphs. Many traditional methods discussed above cannot handle large graphs well because it may take a long time to reach a reasonably low-energy state or because the layout produced may look very cluttered. One effective way for handling large graphs is to eliminate some edges and reduce the graph to a tree, such as a spanning tree [28]. Alternatively, graph layout algorithms can be used to layout the tree and thus determine the positions of graph nodes [35]. Another way is to build a multilevel graph hierarchy to reveal the clustering structure in a graph by coarsening methods. This strategy has been used in [9], [10], [14], [15], [18], [20], [24], [25], [36]. Existing methods vary in how they coarsen a graph, how to derive the initial guess of a layout, and how to refine from the initial layout.

B. Layout Algorithms for Dynamic Graphs

To display dynamic graphs that evolve over time, a good layout should make a good balance between preserving the mental map and achieving good aesthetic quality [5], [6]. Naïvely applying a static graph layout algorithm to graphs of individual time steps often will not preserve the mental map, thus making it difficult for viewers to track the evolution of graphs.

There exist some visual pattern based layouts for dynamic graphs. Yee et al. [42] proposed an animation technique to support interactive graph exploration. They used a radial tree layout and generate animations that linearly interpolate the polar coordinates of the nodes while enforcing ordering and orientation constraints. However, smooth transitions cannot be always expected as changes in the graph may introduce dramatic changes to the hierarchy.

Most other dynamic graph drawing efforts use energy-based layouts. Generally speaking, dynamic graph layout algorithms can be either offline or online, depending on whether the full sequence of graphs is known beforehand or not. For the offline version, it is common to build a global layout for the whole sequence and then derive the layout of each graph from the global layout. For example, in [5], [6], a *super graph* is built as a rough abstraction of the whole sequence of graphs. Every graph in the sequence is a subset of the super graph. For the online version, it is typical to use the layout of one time slice as a starting point to create a new layout for the next time slice, and then further improve the new layout for better aesthetic quality. North [30] proposed an online method for drawing directed acyclic graphs in a hierarchical manner. Brandes and Wagner [1] proposed a Bayesian approach, in combination with force-directed techniques, to generate dynamic graphs. Görg et al. [17] proposed a method for orthogonal and hierarchical graphs. Fisherman and Tal presented an online method [8] for dynamic clustered graphs. They also presented another approach [10] for general graphs. To maintain the mental map, they assigned a movement flexibility degree to each node, allowing the algorithm to focus on nodes that may have large displacements.

C. F+C Techniques for Graph Drawing

F+C techniques have been used for various types of visualization, especially for trees and graphs. This approach displays the foci together with the context which consists of all visual elements or a selected subset of the elements. F+C techniques deal with what elements should be selected to constitute the context, and how the elements should be presented [13]. It is desired to show places near the focal nodes in greater detail while displaying remote regions in successively less detail [12]. Geometric distortion is a typical means to handle the layout in F+C visualization. Based on the visual metaphor of a rubber sheet, these techniques distort the information space using a geometric mapping. As a result, more space is allocated to the foci and nodes nearby, while nodes further away are squeezed. These techniques are exemplified by Sarkar's graphical fisheye [31] and "stretching the rubber sheet" [32], and Gansner et al.'s topological fisheye [15].

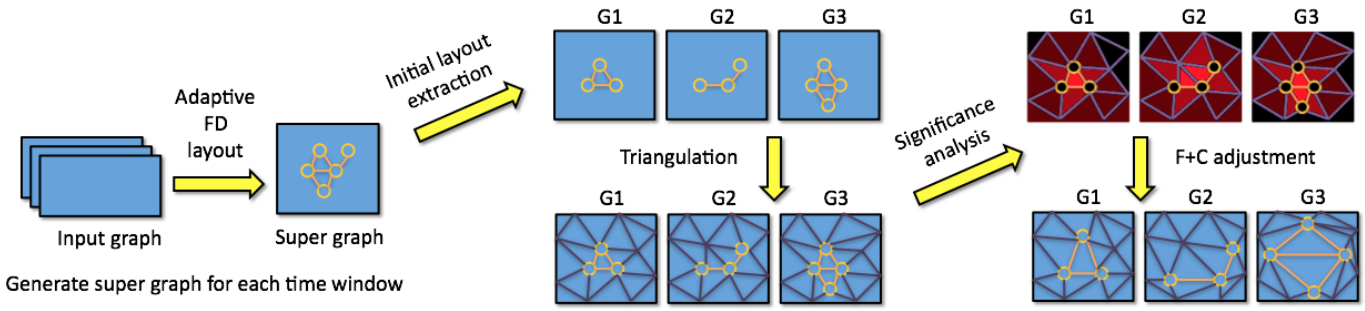
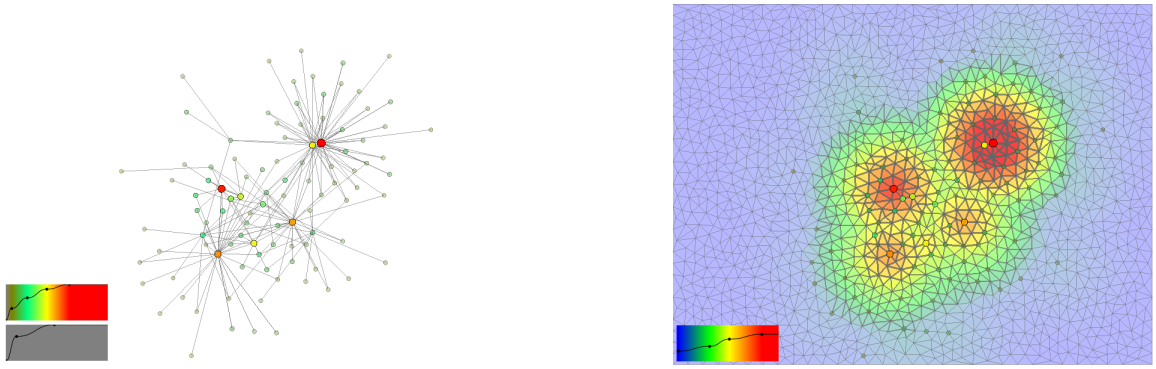


Fig. 1. The overview of our approach for F+C visualization of time-varying graphs. Our approach leverages existing graph layout algorithms to produce initial layouts and performs significance analysis on nodes, faces, and edges of the triangulated version of initial layouts. F+C is achieved through optimization by minimizing an energy function.



(a) (b)

Fig. 2. (a) the importance values of nodes and edges are mapped to their visual properties such as size/thickness, color, and opacity. The two transfer functions around the corner are for node and edge, respectively. (b) the underlying triangle mesh is displayed where we map face importance to color and edge importance to thickness.

III. Our Approach

A. Overview

An overview of our approach is summarized in Figure 1. Our method takes the input graph and applies existing graph layout algorithms to generate the initial layout. To account for temporal coherence, we leverage the idea of super graph [6] to build a sequence of graphs for each time window, from which we extract an initial graph layout for every time step. Inspired by Wang et al. [39], [40], we formulate F+C visualization as a deformation optimization problem, thus allowing the user to magnify the detail in the regions of interest while shrinking the rest of the graph to keep the entire graph displayed on the screen. When expanding the graph, if we only stretch the edges connecting to the nodes of interest, we may not be able to pull those nodes apart as desired. This is because some of the nodes in the spatial neighborhood may not have edges connecting to those nodes that are intended to be extended. To address this issue, we add an intermediate step that triangulates the initial graph into a triangular mesh, and then deforms the mesh to achieve the desired F+C visualization. Since our algorithm produces well-behaved triangulation, the mesh can be effectively deformed in the subsequent step to achieve desired F+C effects. The deformation solves a constrained optimization which minimizes the energy of the graph. The optimization considers three constraints, *aesthetic balance adjustment*, *weighted edge expansion*, and *temporal coherence preservation*, based on the significance analysis of nodes, edges, and faces of the underlying triangle mesh.

In the following discussions, we denote the time-varying graph as $G_t =< V_t, E_t >$ where $t \in T = [1, n]$ represents time, and V_t and E_t are the sets of nodes and edges at time step t, respectively. The node i and the edge connecting nodes i and j at time step t are denoted as $v_{i,t}$ and $e_{ij,t}$, respectively. The face i at time step t in the triangle mesh is denoted as $f_{i,t}$. We assume that each edge $e_{ij,t}$ in the graph carries a weight $w_{e_{ij,t}}$. The edge weights will be utilized to define the importance of nodes. Each node $v_{i,t}$ carries a weight $w_{v_{i,t}}$ indicating its relative importance, and is used to define the importance of faces and edges in the triangle mesh. If no such information is provided, $w_{v_{i,t}}$ will be 1.0 for all the nodes. The reason that we define node importance first and then derive face and edge importance is because node position is essential for determining a graph layout and face and edge are only auxiliary information used in triangle mesh deformation.

B. Initial Layout

As illustrated in Figure 1, our algorithm can work with any existing layout algorithms to set up an initial layout for the graph in every time step. In this paper, we utilize the FR-layout [11] to place the initial graph. To produce a temporally coherence layout, we divide the entire time sequence into a number of time windows, each of which consists of a sequence

of consecutive time steps. For each time window, we utilize the super graph [6] to generate a force-directed layout, from which an initial layout for every time step is extracted.

When partitioning the time sequence into time windows, it can be a simple uniform partition with an equal number of time steps in each window. Or, a better way is to analyze the graph information at each time step and partition the time in a nonuniform way with the nature of the time-varying graph taken into account. Similar to the importance-driven time-varying data visualization work presented in [37], we compute the conditional entropy (Equation 2) for each time step with respect to its neighboring time steps and derive the importance for each time step as follows.

$$I_t = \sum_{k=t-m}^{t-1} w_k H(X_t|Y_k), \tag{1}$$

and

$$H(X|Y) = \sum_{x\in X}\sum_{y\in Y} p(x,y)\log\frac{p(y)}{p(x,y)}, \tag{2}$$

where I_t is the importance value of time step t, m is the window size considered, w_k is the weight associated with time step k. The closer k to t, the larger the weight. $\sum_{k=t-m}^{t-1} w_k = 1.0$. In our case, the entropy is evaluated based on the distribution of node importance defined in Section III-D (Equation 5). That is, $p(x,y)$ is the joint probability of node importance at time steps X and Y, and $p(y)$ is the marginal probability of node importance at time step Y. A higher (lower) importance indicates that the corresponding time step has a higher (lower) degree of change compared with its neighboring time steps, and thus the length of the time window that contains this time step should be smaller (larger). Such a nonuniform partition will evenly distribute the variations of the graphs among all time windows, which makes it more amenable to preserving the temporal coherence for initial layout generation and the subsequent deformation. Note that in our implementation, two consecutive time windows share an overlapped time interval and the time steps falling into the overlapped interval keep the positions of the common nodes the same for both time windows. This is to ensure the continuity between time windows.

C. Graph Triangulation

From the initial graph layout produced for each time step, we use the node positions as the constraint to generate a constrained conforming Delaunay triangulation (CCDT) mesh [33]. Figure 2 shows an example of the resulting triangle mesh. With the CCDT, all the triangles produced are well-behaved, i.e., they have similar areas. Based on the input of desired triangle area and the maximum angle within any triangle, we can produce a suitable number of triangles for our need. Additional vertices, called Steiner points, could be inserted to meet constraints on triangle area and angle. Using the triangle mesh rather than the initial graph for F+C adjustment is more amenable for generating desirable effects while maintaining the spatiotemporal coherence of the time-varying graph.

Our graph is embedded in the triangular mesh where the F+C is performed. Therefore, each node can be still expanded or shrunk accordingly even though it does not have any edge to its spatially adjacent nodes in the original graph. Figure 3 shows two examples of F+C adjustment using initial graph edges and triangle mesh edges respectively. In both examples, using the triangle mesh edges more effectively expands the neighborhood of nodes with higher importance and better maintains the consistency of relative position relationships among nodes. Note that in (b) and (c), there is no adjacent edge between red and orange nodes. In (b), F+C adjustment using the initial graph does not work when two neighboring nodes in the layout are not adjacent in the graph. Therefore, both red and orange nodes are squeezed together in (b) but they are appropriately separated in (c). Similar phenomenon can be found in (e) and (f).

D. Significance Analysis

Node Importance. To allow the users to clearly capture the characteristics of the time-varying graph and achieve desired F+C visualization, we define the *importance* for every node in the graph at each time step. Specifically, we consider two properties for a node: *centrality* and *authority*. The centrality of a node $v_{i,t}$ is defined as

$$\mathscr{C}(v_{i,t}) = \deg(v_{i,t}) = \sum_j \varepsilon_{ij,t}, \tag{3}$$

where $\deg(v_{i,t})$ returns the degree of node $v_{i,t}$. In an undirected graph, it is the number of edges incident to $v_{i,t}$. $\varepsilon_{ij,t} = 1$ iff there is an edge between $v_{i,t}$ and $v_{j,t}$ in the graph; otherwise, $\varepsilon_{ij,t} = 0$. The authority [25] of a node $v_{i,t}$ is defined as

$$\mathscr{A}(v_{i,t}) = \sum_{v_{j,t}\in V_t} w_{e_{ij,t}}^2 \overline{w}_{v_{j,t}}, \tag{4}$$

where $w_{e_{ij,t}}$ is the weight of edge $e_{ij,t}$ in the original graph and $\overline{w}_{v_{j,t}}$ is the average of edge weights incident to node $v_{j,t}$. $v_{i,t}$ and $v_{j,t}$ are connected by edge $e_{ij,t}$. The authority of a node indicates its representativeness. The squared weights for edges give preference to nodes that are very representative of some nodes over those that are moderately representative of all nodes. We use the mean weight to ensure that the most central nodes are also representative of other less central nodes.

Finally, we define the importance of a node $v_{i,t}$ as

$$\mathscr{I}(v_{i,t}) = \alpha\mathscr{C}(v_{i,t}) + \beta\mathscr{A}(v_{i,t}) + \gamma w_{v_{i,t}}, \tag{5}$$

where α, β, and γ are all in $[0,1]$ and $\alpha+\beta+\gamma = 1.0$. $w_{v_{i,t}}$ is the weight of $v_{i,t}$ carried from the input. Note that we only compute the importance for nodes in the original graph. For other pseudo nodes introduced in the triangle mesh, we assign their importance values as zero.

In practice, our deformation is based on the node importance at every time step. We notice that if a node at two consecutive time steps has significantly different importance values, then the resulting deformation would be flickering which hinders user observation. To alleviate this problem, we blend the

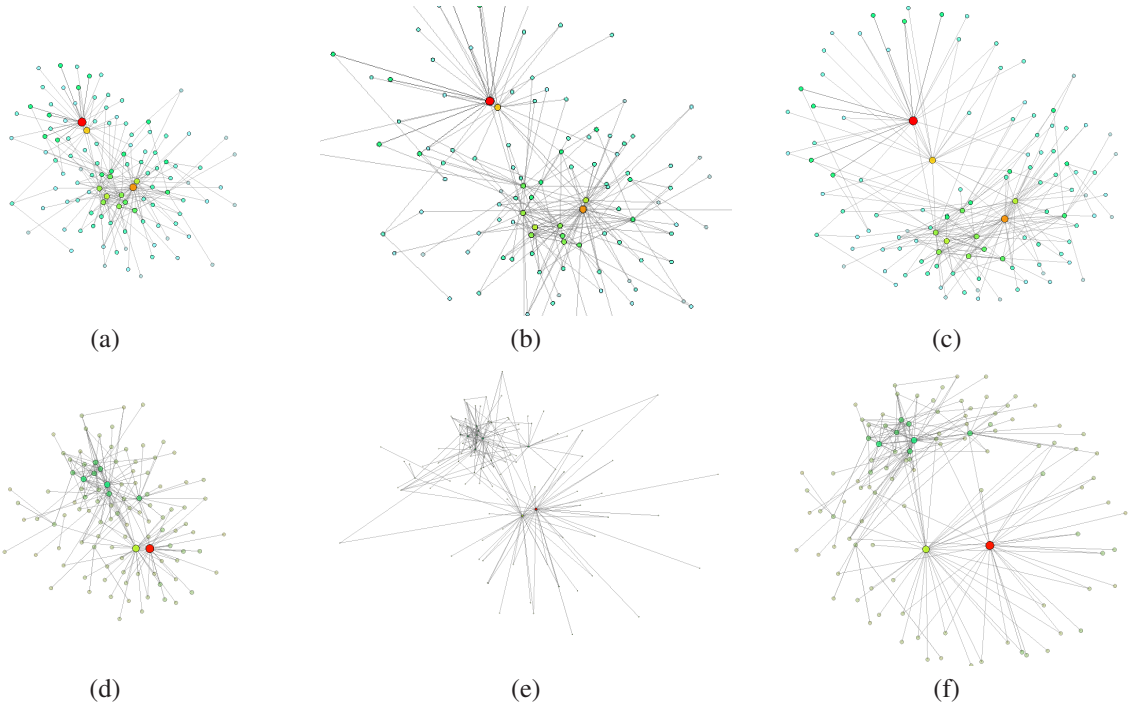


Fig. 3. (a) initial graph (important nodes are colored in red). (b) F+C adjustment using the initial graph edges. (c) F+C adjustment using the triangle mesh edges which allows the nodes close to important nodes to expand as well. (d) initial graph. (e) F+C adjustment using the initial graph edges. (f) F+C adjustment using the triangle mesh edges which avoids the drastic change of relative position relationships among nodes.

importance value of a node with its values at previous $m-1$ time steps

$$\overline{\mathscr{I}}(v_{i,t}) = \sum_{l=t-m+1}^{t} w_l \mathscr{I}(v_{i,l}), \tag{6}$$

where m is the size of blending window, w_l is normalized weight for time step l. The closer l to t, the larger the weight. $\sum_{l=t-m+1}^{t} w_l = 1.0$.

Face Importance. For each face in the triangle mesh, we define its importance as:

$$\mathscr{I}(f_{i,t}) = \max_{v_{j,t} \in V_t} \mathscr{I}(f_{i,t}, v_{j,t}), \tag{7}$$

and

$$\mathscr{I}(f_{i,t}, v_{j,t}) = \begin{cases} 0, & \overline{\mathscr{I}}(v_{i,t}) = 0.0 \\ 0, & \mathrm{dis}(f_{i,t}, v_{j,t}) > 0.5 \times \overline{e} \\ \overline{\mathscr{I}}(v_{j,t}) \frac{\overline{e} - 2 \times \mathrm{dis}(f_{i,t}, v_{j,t})}{\overline{e}}, & \text{otherwise} \end{cases} \tag{8}$$

where $\mathrm{dis}(f_{i,t}, v_{j,t})$ is the distance from $v_{j,t}$ to the center of mass in face $f_{i,t}$ and $\overline{e}$ is the average edge length computed from the original graph.

Edge Importance. With the face importance, the importance of an edge $e_{ij,t}$ in the triangle mesh can be defined as the average of the importance of its incident faces. That is,

$$\mathscr{I}(e_{ij,t}) = \frac{\sum_{f_{k,t} \in F_{e_{ij,t}}} \mathscr{I}(f_{k,t})}{||F_{e_{ij,t}}||}, \tag{9}$$

where $F_{e_{ij,t}}$ is the set of faces incident to edge $e_{ij,t}$. For the triangle mesh, $||F_{e_{ij,t}}||$ is two if $e_{ij,t}$ lies inside of the mesh and is one if $e_{ij,t}$ lies on the mesh boundary.

In Figure 2, we illustrate a graph where we map the importance values of nodes to their size, color, and opacity. More important nodes are drawn with bigger circles and more opaque colors. The thickness, color, and opacity of edges indicate their respective importance values. That is, more important edges are drawn with thicker lines and more opaque colors. The triangle mesh shows the importance of faces and edges. Such a visualization allows important nodes, edges, and faces to stand out as the foci in a visually striking way.

We note that the users can adjust the importance values for nodes or edges during their interaction. For example, the users may choose some nodes as the foci and the importance values of these nodes will increase accordingly to reflect the user preference for the following optimized F+C visualization.

E. Optimized F+C

Based on the result of significance analysis, we will build an *importance map* for every time step of the graph to guide the creation of F+C visualization. The key to achieve smooth F+C visualization lies in maintaining the continuity and relative relations among nodes and edges. We take into account the following conditions and constraints to define our objective energy function.

Aesthetic Balance Adjustment. The use of the force-directed graph in conjunction with the super graph layout produces a single, static graph for all time steps using all nodes and edges in the time-varying graph, in which every time step corresponds to a subset of the super graph. Although it is

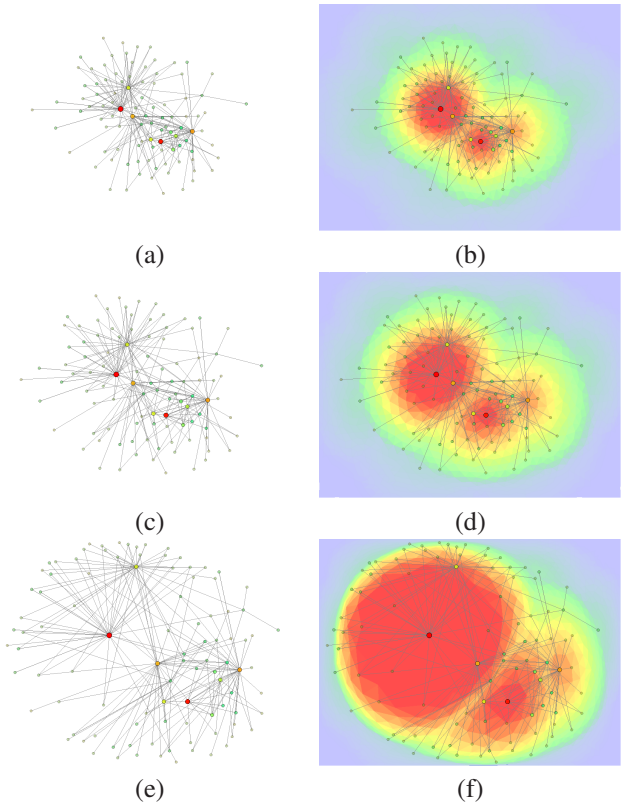


Fig. 4. The effects of aesthetic balance adjustment and weighted edge expansion. (a) shows the graph of a time step extracted from the super graph. (b) is the corresponding triangle mesh of (a). (c) and (d) show the graph and the triangle mesh after aesthetic balance adjustment, respectively. (e) and (f) show the graph and the triangle mesh after weighted edge expansion, respectively. $s = 10$ in this example.

a perfectly temporal coherence solution, the resulting initial graph layout for every time step does not have a good balance between aesthetic quality and dynamic stability. To improve this, we add the aesthetic balance adjustment constraint. The goal is to have the area of each face in the triangle mesh match the its importance value. That is,

$$A_{f_{i,t}} = A \times \frac{\mathscr{I}(f_{i,t})}{\sum_{f_{j,t} \in F} \mathscr{I}(f_{j,t})}, \tag{10}$$

where $A_{f_{i,t}}$ is the area of face $f_{i,t}$, $A = w \times h$ is the area of the drawing region (w and h are the width and height, respectively), and $\mathscr{I}(f_{i,t})$ is the importance of face $f_{i,t}$ (Equation 7). To expand each face to match the desired area $A_{f_{i,t}}$, we adjust each of its edges to an optimal length

$$l(e_{ij,t}) = \sqrt{\frac{4}{\sqrt{3}} A \times \frac{\mathscr{I}(e_{ij,t})}{\sum_{f_{k,t} \in F} \mathscr{I}(f_{k,t})}}, \tag{11}$$

where $\mathscr{I}(e_{ij,t})$ is the importance of edge $e_{ij,t}$ (Equation 9). We now add the following constraint

$$D_a = \sum_{t \in T} \sum_{e_{ij,t} \in E_t} ||e'_{ij,t} - l(e_{ij,t}) \hat{e}_{ij,t}||^2, \tag{12}$$

where $e'_{ij,t}$ is the deformed version of $e_{ij,t}$ and $\hat{e}_{ij,t}$ is the unit vector of $e_{ij,t}$. Figure 4 shows an example graph before ((a) and (b)) and after balance adjustment ((c) and (d)). As we can see, the graph after adjustment introduces dynamic changes to the layout which means the same node in different time steps may not stay at the same position. But the spatiotemporal coherence will still be preserved as our optimization operates on all time steps simultaneously. Expanding triangle faces better utilizes the drawing area and improves the aesthetic quality.

Weighted Edge Expansion. Recall that we compute an importance value for every edge in the triangle mesh, i.e., $\mathscr{I}(e_{ij,t})$ in Equation 9. Let us denote s as a scaling factor given by the user during graph expansion for F+C visualization. In general, we have $s > 1$, and the larger the value of s, the higher the degree of expansion applied to the nodes in the focus. For edges with higher importance values, we need to expand them more compared with edges of lower importance values. In our energy model, we want to minimize the following term related to the graph edges,

$$D_e = \sum_{t \in T} \sum_{e_{ij,t} \in E_t} \mathscr{I}(e_{ij,t}) ||e'_{ij,t} - s_t s(e_{ij,t}) l(e_{ij,t}) \hat{e}_{ij,t}||^2, \tag{13}$$

where

$$s(e_{ij,t}) = 1 + \mathscr{I}(e_{ij,t})^2 (s-1).$$

In Equation 13, $e'_{ij,t}$ is the deformed version of $e_{ij,t}$, $\hat{e}_{ij,t}$ is the unit vector of $e_{ij,t}$, and $l(e_{ij,t})$ is the optimal length of edge $e_{ij,t}$. $s(e_{ij,t})$ is computed according to user-specified s and it indicates the expected scaling factor associated with edge $e_{ij,t}$. s_t is an unknown scaling factor associated with time and is initialized as 1.0. If $s(e_{ij,t})$ is too large, the resulting node position can be placed outside of the drawing area. To avoid this, we adjust s_t to a value less than 1.0 to pull the node position back to stay within the drawing area. If the importance of an edge approaches zero, the edge will keep its original length. Figure 4 shows an example graph before ((a) and (b)) and after ((c) and (d)) edge expansion. It is clear that weighted edge expansion allows edges with higher importance values to expand while edges with lower importance values are shrunk. As a result, nodes with higher importance values are highlighted as nodes with lower importance values are pushed away. In the above F+C scenario, after the user specifies a scaling factor s, our algorithm first computes an initial scaling factor $s(e_{ij,t})$ for each edge, i.e., weighted by its importance, and then computes the optimized scaling value. Another useful F+C scenario is to allow the user to assign the scaling factor s to some nodes of interests directly. Then we assign 1.0, i.e., the maximum importance value, in Equation 6 for each selected node. Finally, we apply the same principle described above to perform F+C visualization. These selected nodes will be expanded and the remaining less important nodes will be shrunk accordingly to achieve desired F+C effects.

Temporal Coherence Preservation. To maintain temporal coherence in the resulting time-varying graph, nodes with higher importance values should keep their locations as close to that of the previous time step as possible. To ensure this, we add the following energy term to make sure that nodes

Algorithm 1 LINEARSYSTEMSOLVER $(\mathbf{A}, \mathbf{V}', B(V))$

Use node positions in the initial layout at each time step as the initial guess for the first iteration
done $\Leftarrow$ **false**{*done* indicates that whether more iterations are needed or not}
while *done* = **false do**
  Use the current iteration result $\mathbf{V}$ to solve the unknown node positions $\mathbf{V}'$ constrained by the boundary and overlapping conditions
  if any node's new position is beyond the drawing region **then**
    Adjust s_t to pull the node position back to stay within the drawing region
  else
    if each node's position change between the current and previous iterations is less than one pixel **then**
      done $\Leftarrow$ **true**
    end if
  end if
  Update the vertex set $\mathbf{V}' = \mathbf{0.7} \times \mathbf{V}' + \mathbf{0.3} \times \mathbf{V}$
end while

in the focus will not move too much accumulatively over the time series:

$$D_t = \sum_{t=1}^{n-1} \sum_{v_{i,t} \in V_t} \overline{\mathscr{I}}(v_{i,t}) ||v'_{i,t} - v_{i,t}||^2, \tag{14}$$

where $\overline{\mathscr{I}}(v_{i,t})$ is the importance of node $v_{i,t}$ and $v'_{i,t}$ is the deformed version of $v_{i,t}$, i.e., $v_{i,t}$'s new position at time step $t+1$.

Boundary Constraint. The boundary constraint states that during the deformation, nodes on the boundary of the drawing area at the previous iteration will be forced to keep their positions on the boundary at the current iteration. That is,

$$v'_{i,t,y} = \begin{cases} 0, & v_{i,t} \text{ is on the top boundary} \\ h-1, & v_{i,t} \text{ is on the bottom boundary} \end{cases} \tag{15}$$

and

$$v'_{i,t,x} = \begin{cases} 0, & v_{i,t} \text{ is on the left boundary} \\ w-1, & v_{i,t} \text{ is on the right boundary} \end{cases} \tag{16}$$

where w and h are the width and height of the drawing area, respectively. Together with s_t in Equation 13, the boundary constraint ensures that no node will go out of bound and the graph will be kept within the rectangular drawing area during F+C adjustment.

Overlapping Constraint. The overlapping constraint states that the positions of nodes in the overlapping portion of two consecutive time windows should remain unchanged. That is,

$$v_{i,t,w_j} = v_{i,t,w_{j+1}}, \tag{17}$$

where v_{i,t,w_j} and v_{i,t,w_j+1} denote the positions of node $v_{i,t}$ in the time windows w_j and w_{j+1}, respectively. This overlapping constraint is to ensure that temporal coherence among nodes between neighboring time windows is preserved.

	Enron	DBLP	Tag (week)	Tag (day)
# nodes	151	873	329	329
# time steps	38	31	52	365
time window size	6	3	1	1
scaling factor s	10	10	10	10
initial layout time	0.5s	3.5s	1.1s	3.0s
mesh computation time	0.6	0.6s	0.8s	5.2s
significance computation time	0.6s	1.5s	1.6s	12.1s
mesh deformation time	4.9s	2.4s	7.6s	74.7s

TABLE I
THE TIMING BREAKDOWN FOR THE THREE TIME-VARYING GRAPH DATA SETS. THE TIME REPORTED IS THE COMPUTATION TIME FOR ALL TIME STEPS.

Objective Energy Function. We define the objective energy function as

$$\underset{V_t}{\operatorname{argmin}} \left(aD_a + bD_e + cD_t \right). \tag{18}$$

where a, b, and c are in $[0,1]$ and $a+b+c = 1.0$. Our goal is to minimize the energy function under the three constraints stated above, and to achieve smooth F+C visualization of the time-varying graph.

To find the minimum of Equation 18, we find the value of $\mathbf{V}'$ for which all of the first partial derivatives (i.e., with respect to each variable $\mathbf{V}'$) of Equations 12 to 14 equal 0. We arrange the unknown variables $\mathbf{V}'$ at the left-hand side and known constants on the right-hand side for each linear system. Then, we can find $\mathbf{V}'$ by solving linear systems of equations $\mathbf{A}\mathbf{V}' = B(\mathbf{V})$, where $\mathbf{A}$ represents the coefficients of unknown node positions, $\mathbf{V}'$ represents the unknown node positions we need to solve for, $\mathbf{V}$ represents the node positions solved in the most recent iteration, and $B(\mathbf{V})$ is a vector function of $\mathbf{V}$. Each element in $B(\mathbf{V})$ is a combination of the elements in $\mathbf{V}$. The dimensions of $\mathbf{A}$, $\mathbf{V}$, and $B(\mathbf{V})$ are $m \times 2n$, $2n \times 1$, and $m \times 1$ respectively, where m is the total number of equations in Equations 12 to 14 and n is the number of nodes. Because the number of equations is much larger than that of the unknown node positions under the three constraints (Equations 15, 16, and 17), we iteratively solve for the unknown node positions in a least square sense as sketched in Algorithm 1. At each iteration, if any node's new position violates three constraints, we update the scaling factor s_t (i.e., $s'_t = 0.9 \times s_t$ in our implementation) to pull the node position back to stay within the drawing region. The process continues until the displacement for any node is less than one pixel in two consecutive iterations. To make sure the minimization eventually converges, we repeatedly update the vertex set $\mathbf{V}' = \mathbf{0.7} \times \mathbf{V}' + \mathbf{0.3} \times \mathbf{V}$ at the end of each iteration. To solve the linear least squares problem in each iteration, we apply the GPU-based conjugate gradient solver of [2] with a multigrid strategy, which is more memory- and time-efficient than a direct solver.

IV. RESULTS

We experimented with three time-varying graphs to demonstrate the effectiveness of our approach. In the following, we

	naïve FD-layout	AB	AB + F+C	AB + TC + F+C
	average node displacement			
Enron	183.96	13.37	36.17	15.61
DBLP	168.84	6.43	15.51	10.05
Tag (week)	237.04	4.01	8.58	2.98
Tag (day)	237.27	2.61	5.67	1.82
	average important node displacement			
Enron	133.69	15.91	42.71	7.38
DBLP	124.37	13.40	28.17	6.64
Tag (week)	147.70	4.89	9.76	2.41
Tag (day)	148.59	2.93	6.70	1.42

TABLE II
COMPARISON OF AVERAGE NODE DISPLACEMENT IN PIXEL FOR ALL NODES AND NODES WITH HIGH IMPORTANCE VALUES (> 0.8). AB: AESTHETIC BALANCE. F+C: FOCUS+CONTEXT. TC: TEMPORAL COHERENCE.

describe our data sets and experimental environment, followed by visualization results gathered from our experiments. For better impression of our method and its results, we refer the reader to the supplementary video [1] of this paper.

A. Data Sets

We acquired three time-varying graph data sets from different applications which we describe in the following.

Enron Email. This data set is provided by the UC Berkeley Enron email analysis project. The data set contains email communication records at Enron over a period of a couple of years. We extracted the company's intra-communication records and built a time-varying graph with each time step corresponding to a month's statistics. This gave us 38 time steps with 151 employees. At each time step, each node in the graph represents an employee and each edge carries a weight indicating the number of communication between two employees over that month.

DBLP Coauthorship. We built this data set from the search results of the DBLP Computer Science Bibliography. We searched one influential author in our field and her coauthors as well as her coauthors' coauthors. We built a time-varying graph with each time step corresponding to a year's statistics. This gave us 31 time steps and a total of 873 authors. At each time step, each node in the graph represents an author and each edge carries a weight indicating the number of publications coauthored by two authors accumulated up to that year. This graph grows as the time step increases.

Astronomy Tag. We built this data set from an online astronomy archive maintained by the NASA and Michigan Tech. Everyday the website archives a new astronomy picture along with a paragraph of explanation and a list of meta-tagged keywords. We extracted all tags during the year of 1998 and built two versions of time-varying graphs, one with each time step corresponding to the statistics of a day, and other to the statistics of a week. This gave us 52 (365) time steps with 329 tags. At each time step, each node in the graph represents a tag and each edge carries a weight indicating the number of concurrence of two tags accumulated up to that day (week). This data set is similar to the DBLP coauthorship data set as it also grows as the time step increases.

[1] In addition to the supplementary video, readers can also find high-resolution of video files at http://graphics.csie.ncku.edu.tw/Time_varying_Graph/.

B. Timing Performance and Displacement Comparison

We utilized a GPU implementation of the concurrent number cruncher (CNC) sparse solver [2] to solve the linear system. The constrained conforming Delaunay triangulation (CCDT) mesh was generated following the work of Shewchuk [33]. All tests were run on a PC with an Intel 2.67GHz CPU, 8GB memory, and an nVidia GTX 295 graphics card. In Table I, we reported the timing breakdown for the three data sets. As we can see, the time to perform mesh deformation dominates the total computation time. For mesh deformation, the main limiting factor for timing is the number of time steps in the data set. This is evident by comparing the performance results on the two versions of the astronomy tag data set.

We set the window size to one for the astronomy tag data set. This is mainly due to the reason that this data set grows as the time step increases. That is, the graph of the current time step is updated from the graph in the previous time step with some new nodes and edges. For this kind of time-varying graph, if we use a window size larger than one, the initial layouts of later time windows will be strongly influenced by the layouts of previous windows (for maintaining temporal coherence between windows). This may lead to undesired layout quality problems for later time windows. The DBLP coauthorship data set also has this issue. But such an influence is not as significant because the number of the time steps in the data set is relatively small.

In Table II, we compare the average of accumulated displacements of nodes for the test data sets. We list two cases: node displacement for all nodes and for only nodes with their importance values greater than 0.8. As we can see, naïve force-directed layout incurs the most node displacement, making it very difficult for users to track changes over time. Our approach produces much smaller node displacement with the addition of aesthetic balance adjustment. Introducing F+C brings larger displacement. However, when temporal coherence is also considered, the average displacement is in general fairly small. Our results accentuate nodes of focus while significantly reducing the average node displacement, thus offering a more engaging experience for users to observe time-varying graphs.

C. Significance Adjustment

We allow the user to adjust the weights for node authority and centrality (Equation 5) to highlight different aspects of the graph. In Figure 5, we show two examples with one favoring node authority ($\alpha = \gamma = 0.5,\ \beta = 0.0$) and the other favoring node centrality ($\beta = \gamma = 0.5,\ \alpha = 0.0$). The initial graph layout is the same while the resulting layouts are different. Adjusting these parameters allows the user to observe different characteristics of the graph accordingly. In addition, blending node importance using a time window can generate smooth layout results over time. The larger the size of the time window, the smoother the resulting time-varying graph. Figure 6 gives such an example to illustrate this effect.

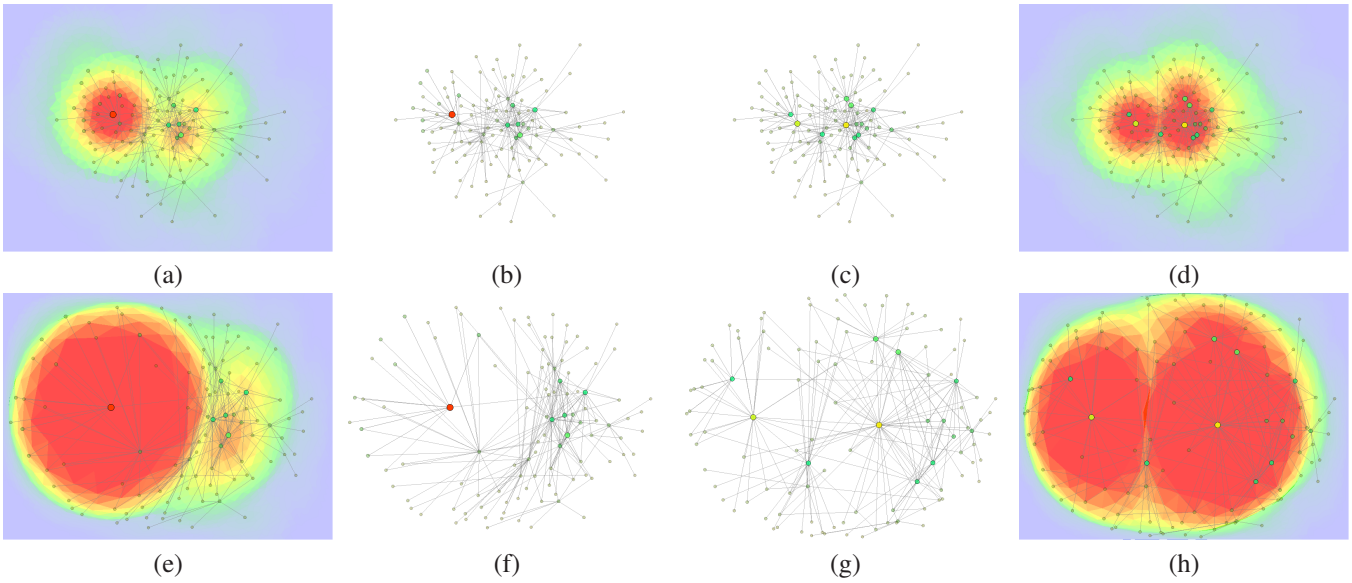


Fig. 5. The same initial graph layout before deformation where node importance is derived by favoring node authority and centrality in (b) and (c) respectively. Their corresponding triangle meshes are displayed in (a) and (d) respectively. (f) and (g) are the adjusted graph layouts for (b) and (c) respectively. (e) and (h) are the corresponding triangle meshes of (f) and (g) respectively.

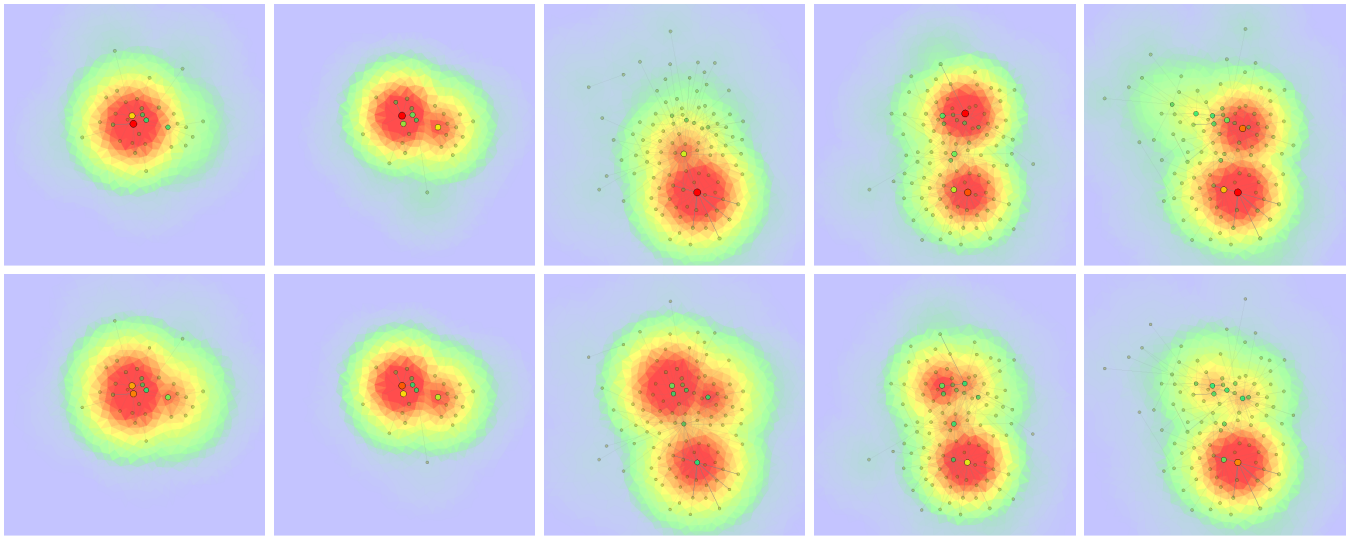


Fig. 6. Top row, left to right: triangle meshes showing the significance of the time-varying graph for five consecutive time steps without blending node importance over time. Bottom row, left to right: triangle meshes showing the significance of the time-varying graph at corresponding time steps with blending node importance. The size of time window is five in this example.

D. Time Budget Allocation

When we have a large number of time steps in a time-varying graph, allocating a given time budget for animation based on the importance of each time step allows us to perform importance-driven time-varying graph visualization. That is, we can slow down the animation when we encounter important time steps (their conditional entropies with respect to neighboring time steps are high), and speed up the animation when we encounter non-important time steps (their conditional entropies with respect to neighboring time steps are low). In our supplementary video, we show a comparison of uniform time budget allocation without considering the importance for each time step and nonuniform time budget allocation with the importance of time steps considered. As we can see from the accompanying video, this importance-driven technique allows us to observe the graph better as more time is spent on the time steps that are more important (i.e., their difference with respect to neighboring time steps are larger thus requiring more time for observation).

E. F+C Visualization

In Figures 7, 8, and 9, we show the comparison of the three data sets with the initial graph layout extracted from the super graph, the adjusted layout, and the final layout after mesh deformation. For better observation, we add halos to nodes with high importance values for highlighting. The area of a

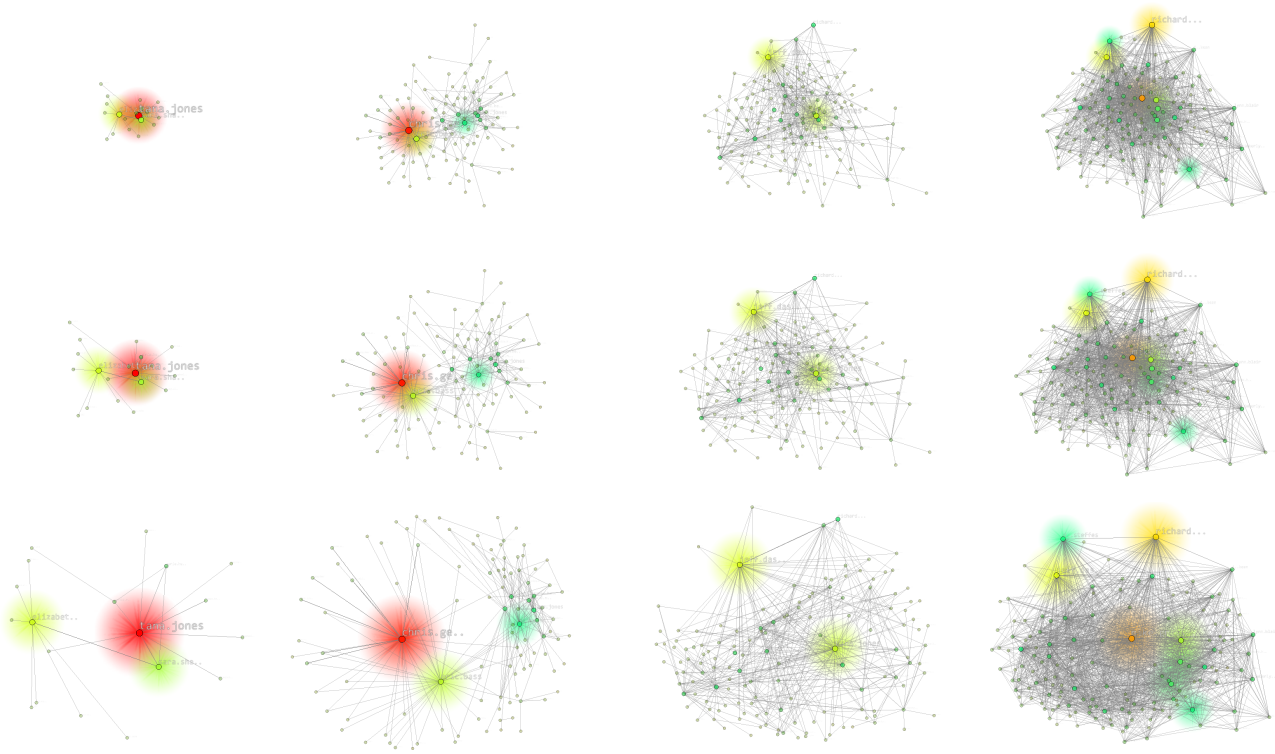


Fig. 7. Left to right are five selected time steps of the Enron email data set. Top to bottom are the initial graph layout extracted from the super graph, the adjusted layout only taking into account aesthetic balance and temporal coherence, and the final layout after mesh deformation based on the significance analysis results, respectively.

halo is proportional to the node's importance value, indicating its significance. Compared with the initial layouts, the final layouts better utilize the screen space and better highlight the significant nodes in the F+C visualization, leading to a more effective way of tracking important nodes over time and a better understanding of the overall time-varying graphs.

Another nice feature we provide is multi-F+C visualization. In this scenario, the user specifies multiple foci in the graph. We update the significance accordingly and produce graph visualization with multiple F+C. Figure 10 shows such an example.

V. Discussion

Our deformation-based optimization framework for time-varying graph visualization is inspired by Wang et al. [38]–[40]. While they solved the F+C problems for static polygon and volume data [39], [40], a direct application of the methods [39], [40] by adding spatiotemporal coherence terms in [38] to time-varying graphs does not lead to smooth F+C visualization. This is because unlike videos frames, which tend to change more smoothly, a time-varying graph can experience more abrupt changes in the locations, sizes, and connectivities of nodes and edges at consecutive time steps. In addition, because spatially adjacent nodes in a graph may not always have edges connecting them, when trying to shrink or expand the graph to achieve F+C views by pulling the graph nodes, we need special treatments beyond what is in [39], [40] to achieve satisfactory results.

Compared to the original super graph algorithms [5], [6], the framework proposed in this paper addresses two major limitations. First, while the super graph algorithms can preserve the mental map, i.e., spatiotemporal coherence, by using the global layout for a given sequence of graphs, they do so at the cost of certain aesthetic criteria. The authors of [5] solved this problem by compromising aesthetic quality and dynamic stability, which, however, is very computationally expensive. Another major limitation is that, when we perform F+C visualization on super graphs, the weights of nodes can change over time, i.e., nodes of focus change in size and location. In this case, it is very difficult to maintain the spatiotemporal coherence of the graphs.

In our framework, a combination of the super graphs and the deformation model in [39], [40] is used to generate smooth F+C visualization for time-varying graphs. The advantage of such a combination is twofold. First, for F+C visualization, the super graphs can maintain spatiotemporal coherence to some extent on several consecutive time steps (i.e., a local time window), thus avoiding nodes to be placed in very distinct locations within the time window. We can maintain high aesthetic quality in the resulting time-varying graph using novel spatiotemporal energy terms in our deformation framework to solve the problem of [6] without incurring high computation cost to generate the graph layout for every time step. As a result, interactive F+C visualization of time-varying graphs becomes possible using our framework. The second advantage is that the energy minimization approach we use can maintain spatiotemporal coherence for graph nodes of

Fig. 8. Left to right are five selected time steps of the DBLP coauthorship data set. Top to bottom are the initial graph layout extracted from the super graph, the adjusted layout only taking into account aesthetic balance and temporal coherence, and the final layout after mesh deformation based on the significance analysis results, respectively.

various weights while our deformation model achieves stable F+C views of the graphs. In addition, the transition between the graphs in consecutive local time windows is delivered smoothly. To the best of our knowledge, using optimization-based methods to generate F+C visualization of time-varying graphs has not been studied previously. Our work naturally integrates dynamic graph drawing and multi-F+C visualization into an optimization framework.

VI. Conclusions and Future Work

Effective techniques for visualizing time-varying graphs are in growing demand. Traditional visualization techniques for time-varying graphs, however, often fail to help users track nodes of interest in complex graphs due to the ineffective graph layouts. In this paper, we propose a novel solution to visualize time-varying graphs that allows users to generate customized layouts and animations via simple user interaction. We achieve this by utilizing the ideas of the super graph and graph triangulation to produce a smooth and spatiotemporally coherent graph visualization with multi-F+C capability. By transforming the graph layout problem to a constrained optimization problem for mesh deformation, we are able to improve the layout directly extracted from the super graph and in the meanwhile, highlight nodes of interest and the associated connections around their neighborhood. Through adjusting the importance of graph nodes, the users can dynamically change the significance distribution in the graph and observe the new layout. The users can also specify different focus nodes at a particular time step and rearrange the graph layout effectively via GPU-accelerated mesh deformation. Importance-driven time budget allocation allows us to produce animations with an emphasis on important time steps to facilitate detailed analysis of the graph data.

Since the definition of importance has a significant impact to the result of mesh deformation, significance analysis of graph plays an important role in the design of graph layout. This paper utilizes the definitions of degree centrality and authority from graph theory to determine the importance of nodes. In addition to what is used in this paper, there exist other ways to define degree centrality such as betweenness centrality, closeness centrality, and eigenvector centrality [41]. All these can be utilized and incorporated into our work to generate desired layout results. We would like to note that the use of mesh deformation has its own limitation. Due to the continuity of mesh, we cannot change the relative node positions in the initial graph layouts. We utilize the spatiotemporal coherence of the super graph to generate an initial layout for every time step. However, if the initial layouts do not exhibit spatiotemporal coherence at all, mesh deformation could lead to undesired layouts by breaking and flipping node relationship in the mesh. For a time-varying graph with a large number of nodes and/or time steps, the optimization for mesh deformation could take a long period of time and the results produced could be very complex. In the future, we would like to improve our algorithm by taking a multi-level approach to prioritize nodes for time-varying graph visualization. This would reduce the complexity of graph and speed up the computation time, providing a more efficient way

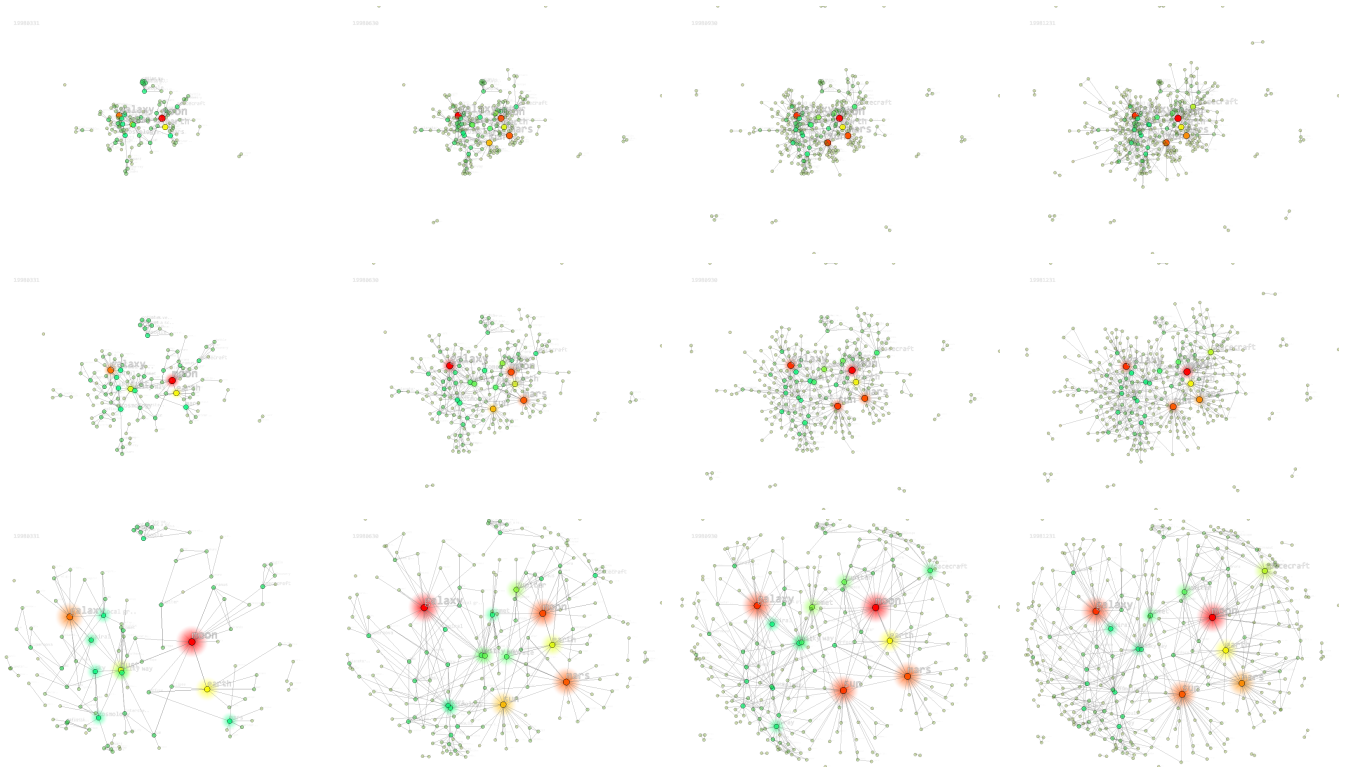


Fig. 9. Left to right are five selected time steps of the astronomy tag data set. Top to bottom are the initial graph layout extracted from the super graph, the adjusted layout only taking into account aesthetic balance and temporal coherence, and the final layout after mesh deformation based on the significance analysis results, respectively.

to visualizing large time-varying graphs.

REFERENCES

[1] U. Brandes and D. Wagner. A Bayesian paradigm for dynamic graph layout. In *Proceedings of International Symposium on Graph Drawing*, pages 236–247, 1997.

[2] L. Buatois, G. Caumon, and B. Lévy. Concurrent number cruncher: A GPU implementation of a general sparse linear solver. *International Journal of Parallel, Emergent and Distributed Systems*, 24(3):205–223, 2009.

[3] J. D. Cohen. Drawing graphs to convey proximity: An incremental arrangement method. *ACM Transactions on Computer-Human Interaction*, 4(3):197–229, 1997.

[4] J. Díaz, J. Petit, and M. Serna. A survey of graph layout problems. *ACM Computing Surveys*, 34(3):313–356, 2002.

[5] S. Diehl and C. Görg. Graphs, they are changing. In *Proceedings of International Symposium on Graph Drawing*, pages 23–30, 2002.

[6] S. Diehl, C. Görg, and A. Kerren. Preserving the mental map using foresighted layout. In *Proceedings of Eurographics - IEEE TCVG Symposium on Visualization*, pages 175–184, 2001.

[7] P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.

[8] Y. Frishman and A. Tal. Dynamic drawing of clustered graphs. In *Proceedings of IEEE Symposium on Information Visualization*, pages 191–198, 2004.

[9] Y. Frishman and A. Tal. Multi-level graph layout on the GPU. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1310–1319, 2007.

[10] Y. Frishman and A. Tal. Online dynamic graph drawing. In *Proceedings of Eurographics - IEEE VGTC Symposium on Visualization*, pages 75–82, 2007.

[11] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software - Practice and Experience*, 21(11):1129–1164, 1991.

[12] G. W. Furnas. Generalized fisheye views. *ACM SIGCHI Bulletin*, 17(4):16–23, 1986.

[13] G. W. Furnas. A fisheye follow-up: Further reflections on focus+context. In *Proceedings of ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 999–1008, 2006.

[14] P. Gajer and S. G. Kobourov. GRIP: Graph drawing with intelligent placement. In *Proceedings of International Symposium on Graph Drawing*, pages 222–228, 2001.

[15] E. Gansner, Y. Koren, and S. North. Topological fisheye views for visualizing large graphs. In *Proceedings of IEEE Symposium on Information Visualization*, pages 175–182, 2004.

[16] E. R. Gansner, Y. Koren, and S. North. Graph drawing by stress majorization. In *Proceedings of International Symposium on Graph Drawing*, pages 239–250, 2005.

[17] C. Görg, P. Birke, M. Pohl, and S. Diehl. Dynamic graph drawing of sequences of orthogonal and hierarchical graphs. In *Proceedings of International Symposium on Graph Drawing*, pages 228–238, 2005.

[18] S. Hachul and M. Jünger. Drawing large graphs with a potential-field-based multilevel algorithm. In *Proceedings of International Symposium on Graph Drawing*, pages 285–295, 2005.

[19] S. Hachul and M. Jünger. An experimental comparison of fast algorithms for drawing general large graphs. In *Proceedings of International Symposium on Graph Drawing*, pages 235–250, 2006.

[20] D. Harel and Y. Koren. A fast multi-scale method for drawing large graphs. *Journal of Graph Algorithms and Applications*, pages 183–196, 2002.

[21] D. Harel and Y. Koren. Graph drawing by high-dimensional embedding. In *Proceedings of International Symposium on Graph Drawing*, pages 299–345, 2002.

[22] I. Herman, G. Melancon, and M. S. Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, 2000.

[23] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, 1989.

[24] Y. Koren, L. Carmel, and D. Harel. ACE: A fast multiscale eigenvectors computation for drawing huge graphs. In *Proceedings of IEEE Symposium on Information Visualization*, pages 137–144, 2002.

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

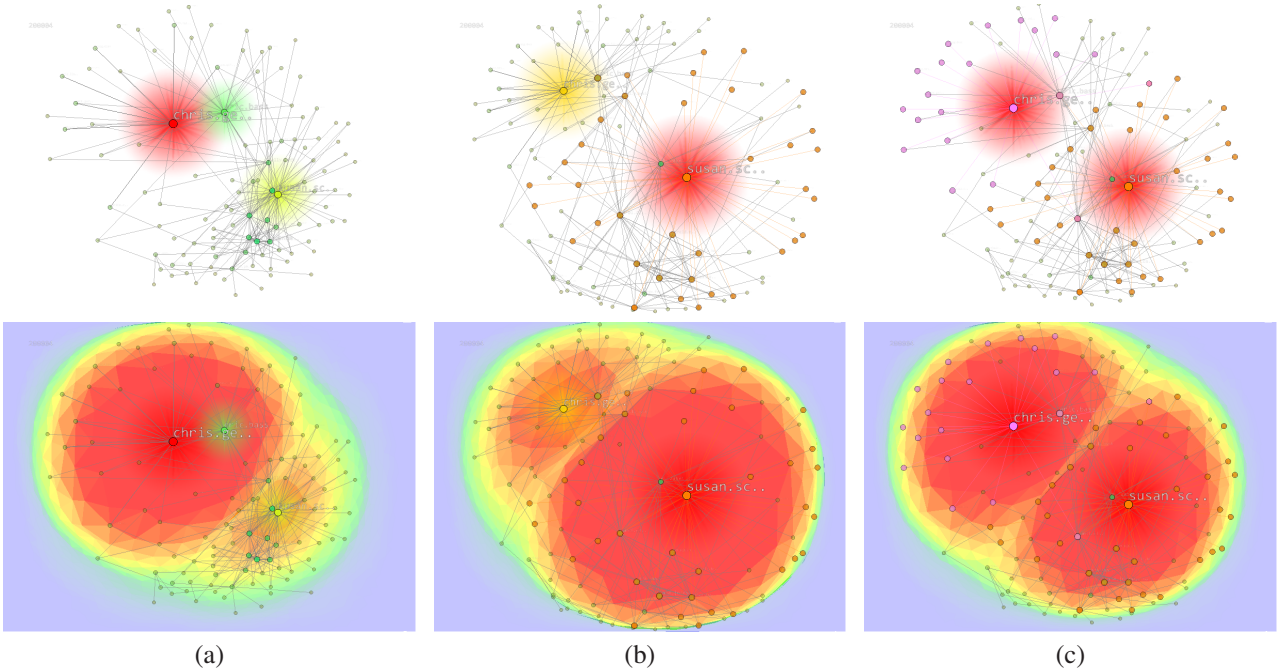


Fig. 10. Multi-F+C Visualization. (a) is the initial graph and its corresponding triangle mesh. (b) is the result with a single focus. (c) is the result with two foci.

[26] C. Muelder and K.-L. Ma. Rapid graph layout using space filling curves. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1301–1308, 2008.

[27] C. Muelder and K.-L. Ma. A treemap based method for rapid layout of large graphs. In *Proceedings of IEEE Pacific Visualization Symposium*, pages 231–238, 2008.

[28] T. Munzner. H3: Laying out large directed graphs in 3D hyperbolic space. In *Proceedings of IEEE Symposium on Information Visualization*, pages 2–10, 1997.

[29] A. Noack. An energy model for visual graph clustering. In *Proceedings of International Symposium on Graph Drawing*, pages 425–436, 2004.

[30] S. C. North. Incremental layout in DynaDAG. In *Proceedings of International Symposium on Graph Drawing*, pages 409–418, 1996.

[31] M. Sarkar and M. H. Brown. Graphical fisheye views of graphs. In *Proceedings of ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 83–91, 1992.

[32] M. Sarkar, S. S. Snibbe, O. J. Tversky, and S. P. Reiss. Stretching the rubber sheet: A metaphor for viewing large layouts on small screens. In *Proceedings of ACM Symposium on User Interface Software and Technology*, pages 81–91, 1993.

[33] J. R. Shewchuk. Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. In *Proceedings of ACM Workshop on Applied Computational Geometry*, pages 203–222, 1996.

[34] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(2):109–125, 1989.

[35] F. van Ham and M. Wattenberg. Centrality based visualization of small world graphs. In *Proceedings of Eurographics - IEEE VGTC Symposium on Visualization*, pages 975–982, 2008.

[36] C. Walshaw. A multilevel algorithm for force-directed graph drawing. In *Proceedings of International Symposium on Graph Drawing*, pages 31–55, 2001.

[37] C. Wang, H. Yu, and K.-L. Ma. Importance-driven time-varying data visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1547–1554, 2008.

[38] Y.-S. Wang, H. Fu, O. Sorkine, T.-Y. Lee, and H.-P. Seidel. Motion-aware temporal coherence for video resizing. *ACM Transactions on Graphics*, 28(5), 2009.

[39] Y.-S. Wang, T.-Y. Lee, and C.-L. Tai. Focus+context visualization with distortion minimization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1731–1738, 2008.

[40] Y.-S. Wang, C. Wang, T.-Y. Lee, and K.-L. Ma. Feature-preserving volume data reduction and focus+context visualization. *IEEE Transactions on Visualization and Computer Graphics*, 2011. To Appear.

[41] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.

[42] K.-P. Yee, D. Fisher, R. Dhamija, and M. Hearst. Animated exploration of dynamic graphs with radial layout. In *Proceedings of IEEE Symposium on Information Visualization*, pages 43–50, 2001.