Coherent Time-Varying Graph Drawing with Multifocus+Context Interaction

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

Abstract—We present a new approach for time-varying graph drawing that achieves both spatiotemporal coherence and multifocus+context visualization in a single framework. Our approach utilizes existing graph layout algorithms to produce the initial graph layout, and formulates the problem of generating coherent time-varying graph visualization with the focus+context capability as a specially tailored deformation optimization problem. We adopt the concept of the super graph to maintain spatiotemporal coherence and further balance the needs for aesthetic quality and dynamic stability when interacting with time-varying graphs through focus+context visualization. Our method is particularly useful for multifocus+context visualization of time-varying graphs where we can preserve the mental map by preventing nodes in the focus from undergoing abrupt changes in size and location in the time sequence. Experiments demonstrate that our method strikes a good balance between maintaining spatiotemporal coherence and accentuating visual foci, thus providing a more engaging viewing experience for the users.

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

1 Introduction

Graph drawing plays an increasingly important role in data understanding for many science and engineering disciplines such as biology, archeology, information retrieval, and VLSI circuit design. More recently, it has 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, existing graph drawing algorithms are primarily focused on static graphs. The more challenging issue of time-varying graph drawing, however, has not received full attention.

Many graphs are dynamic in nature. Examples include event graphs extracted from archives showing event connection and evolution, processor communication graphs obtained from a supercomputer run, and friendship networks inferred from a social website. A critical consideration when designing a time-varying graph layout is to maintain a certain level of spatiotemporal coherence in the visualization of nodes and edges so that their temporal evolution and correlation can be clearly revealed. It is convenient to simply apply a static graph layout algorithm to graphs of individual time steps, either separately or incrementally. However, this treatment cannot guarantee spatiotemporal coherence and a balanced drawing, and hence, the resulting visualization may suffer from undesired artifacts such as flickering or popping (i.e., abrupt changes in the visualization of nodes or edges with respect to size or location). These artifacts make it difficult for viewers to track the changes and thus hinder data understanding.

Another critical consideration for handling time-varying graphs with ever-growing size and complexity is to provide the capability of focus+context (F+C) viewing. F+C visualization stems from the need to show, within a limited display area, both overview (context) and detailed (focus) information simultaneously. Such a capability allows the easy tracking of individual nodes of interest and inferring relationship changes, making it particularly important for the visual analysis of large-scale time-varying graphs through interaction. Although there exist solutions for F+C visualization of static graphs [13], [25] or static data such as polygons [30] or volume data [31], coherent F+C visualization of time-varying graphs has not been fully investigated.

We propose a novel approach for time-varying graph drawing that offers a more engaging viewing experience for users through coherent F+C visualization. Specifically, we formulate the problem of time-varying graph layouting as a deformation optimization problem with an initial layout generated from an existing graph layout algorithm. To generate desired layouts with an F+C effect, we incorporate the concept of the super graph [6] and solve a series of spatiotemporal coherence constraints to preserve coherent contents. Our method allows the users to specify multiple foci in their visualization. We produce a smooth F+C visualization by preventing the nodes in the foci from showing abrupt changes in size and location over time while keeping the context information as stable as possible.

- *K.-C. Feng and T.-Y. Lee are with the Computer Graphics Group/Visual System Lab, Department of Computer Science and Information Engineering, National Cheng Kung University, No. 1 University Road, Tainan 701, Taiwan, ROC. E-mail: stevenf3@gmail.com, tonylee@mail.ncku.edu.tw.*
- *C. Wang is with the Department of Computer Science, Michigan Technological University, Houghton, MI 49931. E-mail: chaoliw@mtu.edu.*
- *H.-W. Shen is with the Department of Computer Science and Engineering, The Ohio State University, 2015 Neil Avenue, Columbus, OH 43210. E-mail: hwshen@cse.ohio-state.edu.*

Manuscript received 17 Aug. 2010; revised 1 May 2011; accepted 4 July 2011; published online 21 July 2011.
Recommended for acceptance by H. Hauser.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2010-08-0189.
Digital Object Identifier no. 10.1109/TVCG.2011.128.

We demonstrate the efficacy of our method with three time-varying graph data sets drawn from different applications.

2 Related Work

Designing effective and efficient graph layouts is one of the central tasks for the graph drawing community. It is also an important topic in information visualization and has been an active area of research for many years. 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 since dynamic graph drawing can often be constructed by leveraging static graph drawing algorithms with additional temporal components.

2.1 Layout Algorithms for Static Graphs

Many previous layout algorithms for static graphs are based on physical analogies such as force or energy. These methods model the graph as a system of physical objects that interact with each other. Optimization techniques are used to minimize the total energy so that the graph converges to an equilibrium state that corresponds to a desired graph layout [1]. Classical examples include the force-directed layout algorithm introduced by Eades [7], the Kamada-Kawai layout [18], and the Fruchterman-Reingold layout [10]. More complex models were also proposed such as LinLog (a clustering energy model) [21] and stress majorization [15]. Researchers also considered node sizes in graph drawing for overlap removal by simply using increased repulsive forces [17] or leveraging the proximity stress model [14].

2.2 Layout Algorithms for Dynamic Graphs

Dynamic graph drawing deals with graphs that evolve over time. To display dynamic graphs, a good layout should strike a good balance among several goals such as preserving the mental map, reusing layouts from previous time steps, and achieving good aesthetic quality [3], [5], [6]. The term *mental map* refers to the abstract structural information a user forms by looking at the graph layout. Misue et al. [20] described three mental map models, i.e., the orthogonal order, proximity, and topology models, that measure the extent by which the graphical attributes have been changed due to a layout adjustment. While two empirical analyses on the mental map by Purchase et al. [23], [24] lead to somewhat contradictory suggestions, we consider that maintaining the mental map and allowing the user to fine tune the degree of mental map preservation are essential. Naïvely applying a static graph layout algorithm to graphs of individual time steps often fails to preserve the mental map well, which makes it difficult for the viewers to track the evolution of graphs.

Generally speaking, dynamic graph layout algorithms can be either *offline* where the full sequence of graphs is known beforehand, or *online* where the full graph sequence is not known in advance. 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, Diehl et al. [5], [6] built a *super graph* as a rough abstraction of the whole graph sequence. 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. Solutions have been proposed for drawing online directed acyclic graphs [22], dynamic clustered graphs [8], and orthogonal and hierarchical graphs [16]. Brandes and Wagner [2] introduced a Bayesian approach, in conjunction with force-directed techniques, to generate online dynamic graphs. Frishman and Tal [9] presented an efficient GPU-based solution to compute stable and aesthetic layouts for online dynamic graphs. To maintain the mental map, they assigned a movement flexibility degree to each node so that nodes with large displacement are focused.

2.3 Focus+Context Techniques for Graph Drawing

F+C techniques have been used for various types of visualization including trees and graphs. This approach displays the foci together with the context which consists of all visual elements or a selected subset of elements. F+C techniques deal with what elements should be selected to constitute the context and how the elements should be presented [12]. It is desired to show places near the focal nodes in greater detail while displaying remote regions in successively less detail [11]. 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 [25] and "stretching the rubber sheet" [26], and Gansner et al.'s topological fisheye [13].

2.4 Our Contribution

To achieve F+C visualization, current techniques make use of distortion either in the geometry space by stretching some edges and shortening others, or in the image space by fisheye transformations and the like. Yet, both have disadvantages: the geometric distortion cannot push apart unconnected nodes, and the image distortion cannot guarantee to preserve edge drawing styles, such as orthogonal drawings. We present a new way to achieve both by distorting not the graph layout itself, but a triangulated, meshed version of it in the geometry space.

In our approach, the user interactively determines one or multiple foci in the graph for dynamic F+C visualization via optimized deformation. We maintain the mental map of time-varying graphs while providing the flexibility to fine tune the degree of mental map preservation so that different viewers can adjust according to their preferences for effective observation. Unlike typical fisheye techniques, by performing a globally optimized deformation of the entire graph, our F+C technique can effectively expand the graph to occupy the available drawing region. Another distinction is that our approach can well preserve the overall graph structure by maintaining relative relationships among important nodes regardless whether they are in the focus or not while squeezing regions of low importance as much as possible. To the best of our knowledge, our work is the first that addresses the issue of multiple F+C visualization

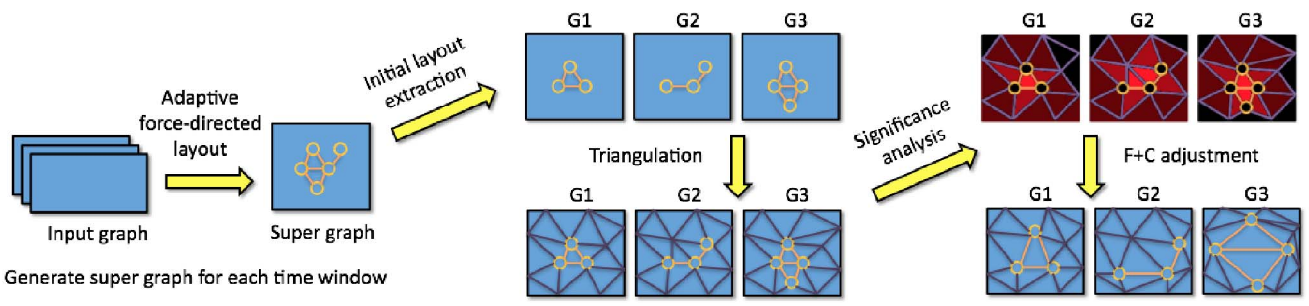

Adaptive force-directed layout
Input graph
Super graph
Generate super graph for each time window
Initial layout extraction
G1
G2
G3
Triangulation
Significance analysis
F+C adjustment

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

in offline, dynamic graph drawing. We point out that due to the use of a time window for the super graph generation, our approach is also suitable for online dynamic graph drawing, provided that the time steps within the sliding time window can be cached during the online processing.

3 Our Approach

3.1 Overview

We sketch an overview of our approach in Fig. 1. Our method takes the input graph and applies an existing graph layout algorithm to generate the initial layout. To account for temporal coherence, we leverage the idea of the 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. [30], [31], we formulate F+C visualization as a deformation optimization problem, thus allowing the user to magnify details in regions of interest while shrinking the rest to keep the entire graph displayed on the screen. There is an issue when expanding the graph: if we only stretch the edges connecting 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 expanded. To address this issue, we add an intermediate step that triangulates the initial graph into a triangle mesh, and then deforms the mesh to achieve the desired F+C visualization. The deformation solves a constrained optimization based on the significance analysis of nodes, edges, and faces of the underlying triangle mesh to minimize the energy of the graph.

In the following discussion, we denote the time-varying graph as $G_t = <V_t, E_t>$ where $t \in T = [1, n]$ represents the time step, 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 $f_{i,t}$. In the *original graph*, we assume that each edge $e_{ij,t}$ carries a weight $we_{ij,t}$. We also assume that each node $v_{i,t}$ carries a weight $wv_{i,t}$. If no such information is provided, $wv_{i,t}$ is 1 for all the nodes. Both $we_{ij,t}$ and $wv_{i,t}$ are used to define the importance of nodes in the *triangle mesh*. We further derive the importance of faces and edges accordingly from the importance of the nodes. Note that weights are associated with nodes and edges in the original graph while importance values are associated with nodes, faces, and edges in the triangle mesh. We opt to define node importance first and then derive face and edge importance. The rationale is that node positions are essential for determining a graph layout and faces and edges are auxiliary information used in the triangle mesh deformation.

3.2 Initial Layout

Our algorithm starts with an existing layout algorithm to set up an initial layout for the graph in every time step. In this paper, we utilize the Fruchterman-Reingold layout [10] to generate the initial graph. To produce a temporally coherent layout, we divide the entire time sequence into a number of time windows where each window consists of several 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.

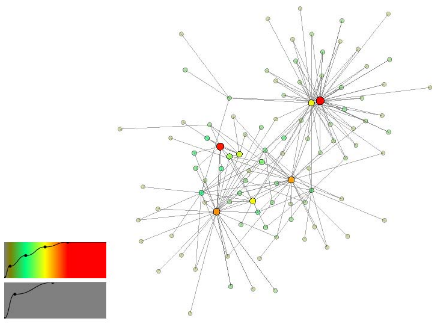
To create the time window, we can simply partition the time sequence uniformly. Another way is to analyze the graph information at each time step and partition the time sequence nonuniformly by taking into account the nature of the time-varying graph. Similar to the importance-driven time-varying data visualization work presented by Wang et al. [28], we compute the conditional entropy (2) for each time step with respect to its neighboring time steps and derive the importance value for each time step

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

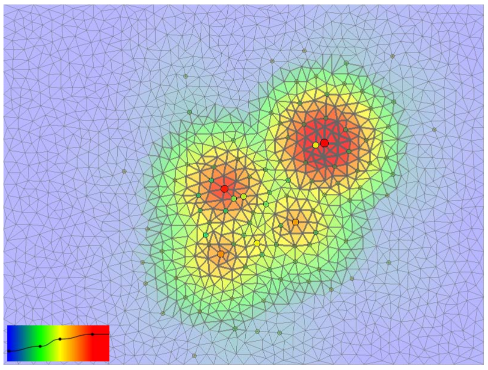
and

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

where I_t is the importance value of time step t, m is the window size considered, and w_k (in $[0,1]$) is the weight associated with time step k. The closer k to t, the larger the weight. $\sum_{k=t-m}^{t-1} w_k = 1$. In our case, the entropy is evaluated based on the distribution of node importance defined in Section 3.4 (5). In (2), $p(x,y)$ is the joint probability of node importance at time steps of x and y, and $p(y)$ is the marginal probability of node importance at the time step of y. A higher (lower) importance indicates a higher (lower) degree of change compared with its neighboring time steps, and thus the corresponding length of the time window should be smaller (larger). Such a nonuniform partition evenly



(a)



(b)

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

distributes the variation of the graphs among all time windows, which makes it more amenable to preserve the temporal coherence for the initial layout generation and subsequent deformation. To ensure the continuity between time windows, we let two consecutive time windows share an overlapping time interval and the time steps falling into the interval keep their common nodes in the same positions. Fig. 3 illustrates such an example.

3.3 Graph Triangulation

From the initial graph layout produced for each time step, we use the node positions to generate a constrained conforming Delaunay triangulation (CCDT) mesh [27]. Fig. 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 determine a suitable number of triangles accordingly. Additional vertices, called Steiner points, could be inserted to meet the constraints of triangle area and angle. Using the triangle mesh rather than the initial graph for F+C adjustment enables us to generate desirable effects while maintaining a satisfying level of spatiotemporal coherence of the time-varying graph.

Since our graph is embedded in the triangle mesh where the F+C distortion is performed, each node can still be expanded or shrunk even though it is not connected to its spatially neighboring nodes in the original graph. Fig. 4 shows two examples of an F+C adjustment using initial graph edges and triangle mesh edges, respectively. In both examples, using triangle mesh edges more effectively expands the neighborhood of nodes with higher importance and better maintains the consistency of relative positions among nodes. Note that in Figs. 4b and 4c, there is no adjacent edge between the red and orange nodes. In Fig. 4b, F+C adjustment using the initial graph does not work when two neighboring nodes in the layout are not adjacent. Therefore, both the red and orange nodes are squeezed together in Fig. 4b but they are properly separated in Fig. 4c. We can observe similar results in Figs. 4e and 4f.

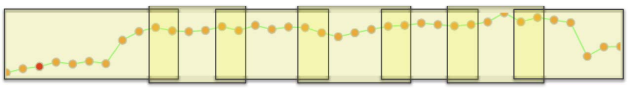


Fig. 3. An illustration showing the importance values of individual time steps as dots and the nonuniform partition of the entire time sequence into seven overlapping time windows. The horizontal direction is for time step and the vertical direction is for importance value. The size of overlapping time intervals is 2 in this example.

3.4 Significance Analysis

3.4.1 Node Importance

To allow the users to clearly capture the characteristics of the time-varying graph and achieve a 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

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

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 [19] of a node $v_{i,t}$ is defined as

$$\mathcal{A}(v_{i,t}) = \sum_{v_{j,t} \in V_t} we_{ij,t}^2 \overline{w}v_{j,t}, \tag{4}$$

where $we_{ij,t}$ 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

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

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

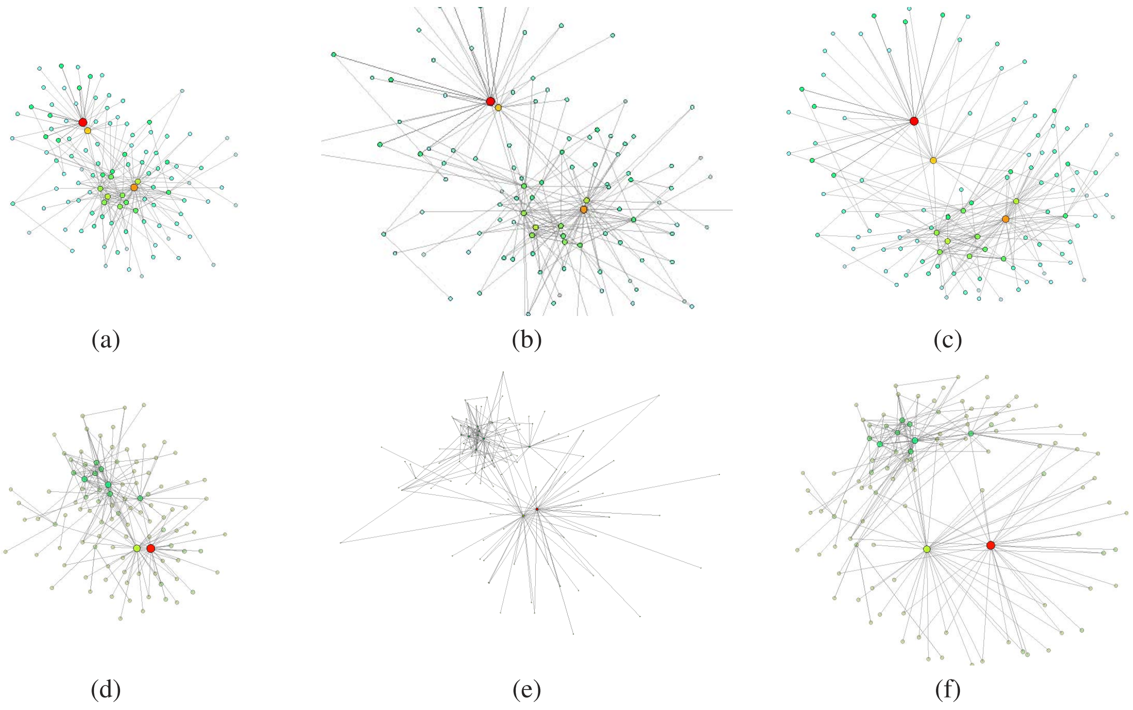


Fig. 4. (a) Initial graph (important nodes shown in red). (b) F+C adjustment using the initial graph edges. (c) F+C adjustment using the triangle mesh edges which allows the nodes close to important nodes to expand as well. (d) Initial graph. (e) F+C adjustment using the initial graph edges. (f) F+C adjustment using the triangle mesh edges which avoids the drastic change of relative positions among nodes.

In practice, our deformation is based on node importance values 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. To alleviate this problem, we blend the importance value of a node with its values at previous $m-1$ time steps

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

where m is the size of blending window, w_l (in $[0,1]$) is the normalized weight for time step l. The closer l to t, the larger the weight. $\sum_{l=t-m+1}^{t} w_l = 1$.

3.4.2 Face Importance

For each face in the triangle mesh, we define its importance as

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

and

$$\mathcal{I}(f_{i,t}, v_{j,t}) = \begin{cases} 0, & \overline{\mathcal{I}}(v_{j,t}) = 0, \\ 0, & \perp(v_{j,t}, f_{i,t}) > \overline{e}/2, \\ \overline{\mathcal{I}}(v_{j,t})\left(1 - \frac{\perp(v_{j,t}, f_{i,t})}{\overline{e}/2}\right), & \text{otherwise}, \end{cases} \tag{8}$$

where $\perp(v_{j,t}, f_{i,t})$ 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. The rationale for (8) is that we only consider the face importance for a contributing node if the node's importance is nonzero and the distance from the node to the center of mass of the face is sufficiently small.

3.4.3 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,

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

where $Fe_{ij,t}$ is the set of faces incident to edge $e_{ij,t}$. For the triangle mesh, $\|Fe_{ij,t}\|$ is 2 if $e_{ij,t}$ lies inside of the mesh and 1 if $e_{ij,t}$ lies on the mesh boundary.

In Fig. 2, we illustrate a graph where we map the importance value of nodes/edges to their sizes/thicknesses, colors, and opacities. More important nodes are drawn with bigger circles and more opaque colors. 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. The users can adjust the importance values for nodes or edges during interaction. For example, the users may choose some nodes as the foci and the importance values of these nodes get increased to reflect user preference for the following optimized F+C visualization.

3.5 Optimized F+C Visualization

The result of the significance analysis guides the following F+C visualization. The key to achieve a smooth F+C visualization lies in maintaining the continuity and relative relationships among nodes and edges. We take into account the following conditions and constraints to define our objective energy function.

3.5.1 Aesthetic Balance Adjustment

Although using the super graph gives a convenient solution that achieves temporal coherence, the resulting initial graph layout for every time step may not have a good balance between aesthetic quality and dynamic stability. To improve this, we add the following constraint to let the area of each face in the triangle mesh match its importance

$$Af_{i,t} = A\frac{\mathcal{I}(f_{i,t})}{\sum_{f_{j,t}\in F_t}\mathcal{I}(f_{j,t})}, \tag{10}$$

where $Af_{i,t}$ 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 $\mathcal{I}(f_{i,t})$ is the importance of face $f_{i,t}$ (7). To expand each face to match the desired area $Af_{i,t}$, we adjust each of its edges to an optimal length

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

where $\mathcal{I}(e_{ij,t})$ is the importance of edge $e_{ij,t}$ (9). In (11), we want each triangle area to match an optimal area $Af_{i,t}$, assuming it is an equilateral triangle. We can add a constraint to approach an equilateral triangle when generating the CCDT mesh. In addition, since each edge $e_{ij,t}$ can be shared by one or two triangles, we compute its optimal length weighted by $\mathcal{I}(e_{ij,t})$, i.e., the average importance of adjacent faces, in (11). 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 $e_{ij,t}$. Fig. 5 shows an example graph before and after aesthetic balance adjustment. The graph after adjustment introduces dynamic layout changes which means that the same node in different time steps may not stay at the same position. But the spatiotemporal coherence is still preserved as our optimization operates on all time steps in each time window simultaneously. Expanding triangle faces better utilizes the drawing area and allows us to improve the aesthetic quality.

3.5.2 Weighted Edge Expansion

Recall that we compute the importance value for every edge in the triangle mesh, i.e., $\mathcal{I}(e_{ij,t})$ in (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 with lower importance values. In our energy model, we want to minimize the following term related to the graph edges:

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

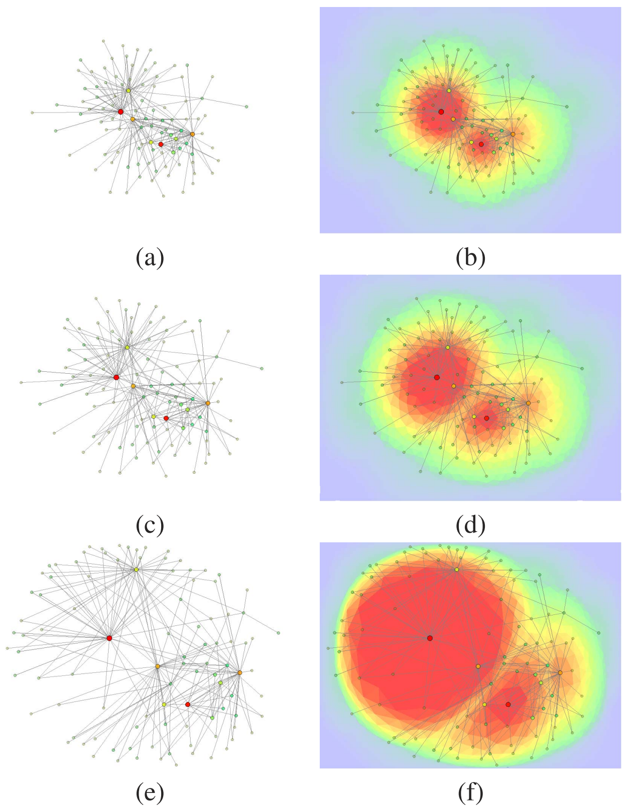


Fig. 5. (a) shows the graph of a time step extracted from the super graph. (b) is the corresponding triangle mesh of (a). (c) and (d) are the graph and the triangle mesh after aesthetic balance adjustment, respectively. (e) and (f) are the graph and the triangle mesh after weighted edge expansion, respectively. $s = 10$ in this example.

where

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

In (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 $e_{ij,t}$. $s(e_{ij,t})$ indicates the expected scaling factor associated with edge $e_{ij,t}$ and is computed according to user specified s. s_t is an unknown scaling factor associated with time and is initialized as 1. If $s(e_{ij,t})$ is too large, the resulting node position can be placed outside of the drawing area. We can avoid this by adjusting s_t to a value less than 1. If the importance of an edge approaches zero, the edge keeps its original length. Fig. 5 shows an example graph before and after weighted edge expansion. It is clear that the 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 aside.

In the above F+C scenario, after the user specifies the 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 interest directly. Then, we assign 1, i.e., the maximum importance value, in (6) for each selected node. Finally, we apply the same principle described above to perform F+C visualization.

3.5.3 Temporal Coherence Preservation

To maintain temporal coherence in the resulting time-varying graph, nodes with higher importance values should keep their locations as stable as possible. We add the following energy term to make sure that nodes in the focus do not move too much accumulatively over the time series

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

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

3.5.4 Boundary Constraint

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

$$v'_{i,t,y} = \begin{cases} 0, & v_{i,t,y} = 0, \\ h-1, & v_{i,t,y} = h-1, \end{cases} \tag{15}$$

and

$$v'_{i,t,x} = \begin{cases} 0, & v_{i,t,x} = 0, \\ w-1, & v_{i,t,x} = w-1, \end{cases} \tag{16}$$

where $v'_{i,t,x}$ $(v_{i,t,x})$ and $v'_{i,t,y}$ $(v_{i,t,y})$ are the x- and y-coordinates of node $v'_{i,t}$ $(v_{i,t})$, respectively, w and h are the width and height of the drawing area, respectively. Together with s_t in (13), this boundary constraint ensures that no node goes out of bound and the graph is kept within the rectangular drawing area during F+C adjustment.

3.5.5 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. Our layout optimization operates on each time window one by one. This overlapping constraint is to ensure that temporal coherence among nodes between neighboring time windows is preserved.

3.5.6 Objective Energy Function

We define the objective energy function as

$$\text{argmin}_{V_t}\big(c_1 D_a + c_2 D_e + c_3 D_t\big), \tag{18}$$

where c_1, c_2, and c_3 are all in $[0, 1]$ and $c_1 + c_2 + c_3 = 1$. Our goal is to minimize the energy function under the three constraints stated above, and to achieve a smooth F+C visualization of the time-varying graph. As sketched in Algorithm 1, we iteratively solve for the unknown node positions $\mathbf{V}'$ in a least-squares sense, where $\mathbf{A}$ represents the coefficients of unknown node positions, $\mathbf{V}$ represents the node positions solved in the most recent iteration, and $B(\mathbf{V})$ is a vector function of $\mathbf{V}$. In practice, we set $c = 0.7$ which produces good results for all graphs we experimented with. To solve the linear least-squares problem, we apply the GPU-based conjugate gradient solver [4] with a multigrid strategy, which is more memory and time efficient than a direct solver.

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

```
 1: Use node positions in the initial layout at each time step
    as the initial guess for the first iteration
 2: done ⇐ false{done indicates whether more iterations
    are needed or not}
 3: while done = false do
 4:    Use the current iteration result V to solve the
       unknown node positions V' constrained by the
       boundary and overlapping conditions
 5:    if any node's new position is beyond the drawing
       region then
 6:       Adjust $s_t$ to pull the node position back to stay
          within the drawing region
 7:    else
 8:       if each node's position change between the
          current and previous iterations is less than one
          pixel then
 9:          done ⇐ true
10:       end if
11:    end if
12:    $\mathbf{V}' = c \times \mathbf{V}' + (1 - c) \times \mathbf{V}$
13: end while
```

4 RESULTS

We experimented with three time-varying graphs to demonstrate the effectiveness of our approach. In the following, we describe our data sets and test environment, followed by visualization results. For a better impression of our method and results, we refer the readers to the supplementary video, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2011.128. In addition, the readers can find high-resolution video clips at http://graphics.csie.ncku.edu.tw/Time_varying_Graph/.

4.1 Data Sets

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

4.1.1 Enron E-Mail

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

4.1.2 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 one year's statistics. This

TABLE 1
The Timing of the Three Time-Varying Graph Data Sets

	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

The time reported is the computation time for all time steps.

gave us 31 time steps and a total of 873 authors. At each time step, each node represents an author and the weight of each edge is the number of publications coauthored by the two authors accumulated up to that year. This graph grows as the time step increases.

4.1.3 Astronomy Tag

We built this data set from an astronomy archive maintained by NASA and Michigan Tech. Everyday, the website features a new astronomy picture along with a paragraph of explanation and a list of metatagged keywords. We extracted all tags during the year 1998 and built two time-varying graphs where each time step corresponds to the statistics of one day (week). This gave us 365 (52) time steps with 329 tags. At each time step, each node represents a tag and the weight of each edge is the number of co-occurrence of the two tags accumulated up to that day (week).

4.2 Timing Performance and Displacement Comparison

We utilized a GPU implementation of the concurrent number cruncher (CNC) sparse solver [4] to solve the linear system. The CCDT mesh was generated following the work of Shewchuk [27]. All tests were run on a PC with an Intel 2.67 GHz CPU, 8 GB memory, and an nVidia GTX 295 graphics card. In Table 1, we report 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 is the number of time steps. This is evident by comparing the performance results for the two versions of the astronomy tag data set.

We set the window size to 1 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 newly added nodes and edges. For this kind of time-varying graph, if we use a window size larger than 1, the initial layouts of later time windows are strongly influenced by the layouts of previous windows (due to the need to maintain temporal coherence between windows). This may lead to an undesired layout quality 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 time steps is relatively small.

In Table 2, we compare the averages of accumulated displacements for all nodes and for nodes with importance values larger than 0.8. As we can see, the naïve Fruchterman-Reingold layout incurs the most node displacement, making it very difficult for users to track changes. The incremental

TABLE 2
Comparison of Average Node Displacement (in Pixel) for All Nodes and Nodes with High Importance Values (>0.8)

	naïve FR-layout	incremental FR-layout	AB	AB + F+C	AB + F+C + TC
	average node displacement				
Enron	183.96	118.29	13.37	36.17	15.61
DBLP	168.84	60.49	6.43	15.51	10.05
Tag (week)	237.04	55.25	4.01	8.58	2.98
Tag (day)	237.27	44.94	2.61	5.67	1.82
	average important node displacement				
Enron	133.69	67.31	15.91	42.71	7.38
DBLP	124.37	39.50	13.40	28.17	6.64
Tag (week)	147.70	31.84	4.89	9.76	2.41
Tag (day)	148.59	22.12	2.93	6.70	1.42

FR-layout: Fruchterman-Reingold layout. AB: Aesthetic balance. F+C*: focus+context. TC: temporal coherence.*

Fruchterman-Reingold layout, where node positions in the previous frame are used as the input to decide node positions in the current frame, reduces the node displacement substantially. Our approach produces an even smaller node displacement with the addition of aesthetic balance adjustment. Introducing F+C distortion brings larger displacements. However, when temporal coherence is also considered, the average displacement is fairly small in general. Our results accentuate nodes in the focus and significantly reduce the average node displacement, thus offering a more engaging experience for users.

4.3 Significance Adjustment

We allow the user to adjust the weights for node authority and centrality (5) to highlight different aspects of the graph. In Fig. 6, 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. Fig. 7 gives such an example.

4.4 Time Budget Allocation

When a time-varying graph consists of a large number of time steps, we can perform importance-driven time-varying graph visualization by allocating a given time budget for animation based on importance values of time steps. 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 nonimportant time steps. In the supplementary video, which can be found on the Computer Society Digital Library, we show a side-by-side comparison between uniform and importance-driven time budget allocation. As we can see, this importance-driven technique allows us to observe the graph better as more animation time is spent on more important time steps (i.e., their difference with respect to neighboring time steps is larger thus demanding more animation time for clear observation).

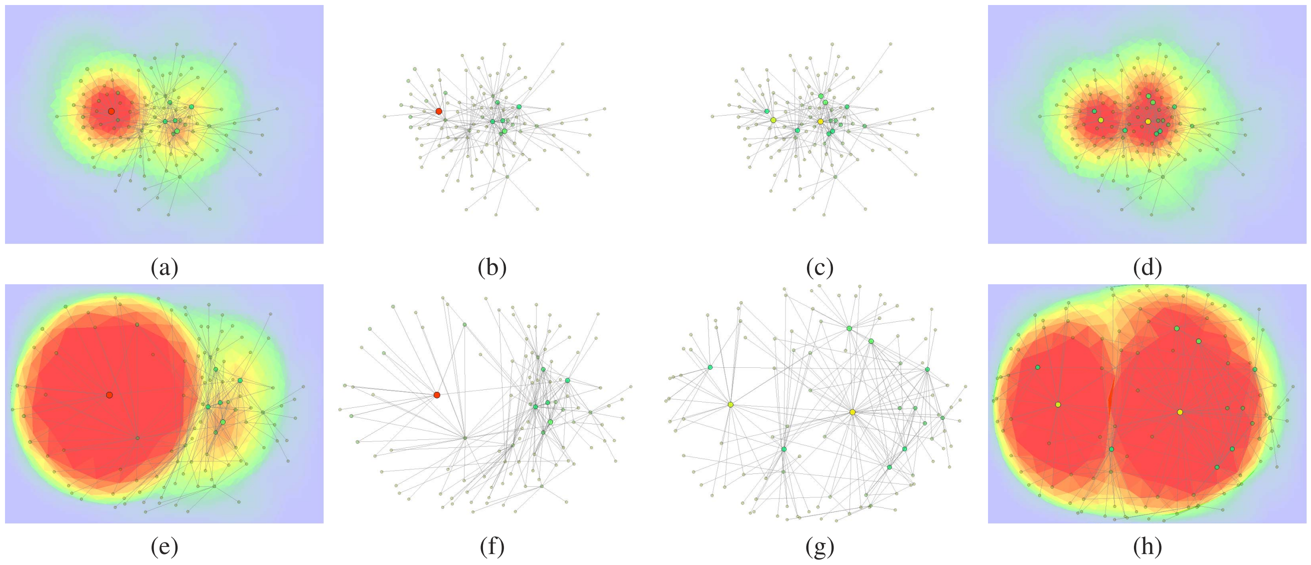


Fig. 6. The same initial graph layout before deformation where node importance is derived by favoring node authority and centrality in (b) and (c), respectively. Their corresponding triangle meshes are displayed in (a) and (d), respectively. (f) and (g) are the adjusted graph layouts for (b) and (c), respectively. (e) and (h) are the corresponding triangle meshes of (f) and (g), respectively.

4.5 F+C Visualization

In Fig. 8, we show the comparison of the astronomy tag data set with the initial graph layout extracted from the super graph, the adjusted layout, and the final layout after mesh deformation. For better observation, we highlight important nodes with a halo. The area of a 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 to 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. Figs. 9 and 10 show additional results with the other two data sets. 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 foci. Fig. 11 shows such an example.

4.6 Mental Map Preservation

Our framework allows the user to fine tune the three parameters in (18) (c_1 for aesthetic balance, c_2 for F+C visualization, and c_3 for temporal coherence) to adjust the degree of mental map preservation. In the supplementary video, we show a comparison among low, medium, and high degrees of mental map preservation with $(c_1, c_2, c_3 = 0, 1, 0)$, $(c_1, c_2, c_3 = 1, 0, 0)$, and $(c_1, c_2, c_3 = 0, 0, 1)$, respectively. We also show our default setting with $(c_1, c_2, c_3 = 1/3, 1/3, 1/3)$. The average node displacement quantifies the difference among these cases with a lower displacement corresponding to a higher mental map. In addition, we compare two sets of parameter settings in the video to show the flexible control over the degree of mental map preservation. Our results confirm that in general, the two extreme cases (i.e., fairly low or high mental maps) are not the best choices for time-varying graph drawing. We recommend the default setting with $(c_1, c_2, c_3 = 1/3, 1/3,$

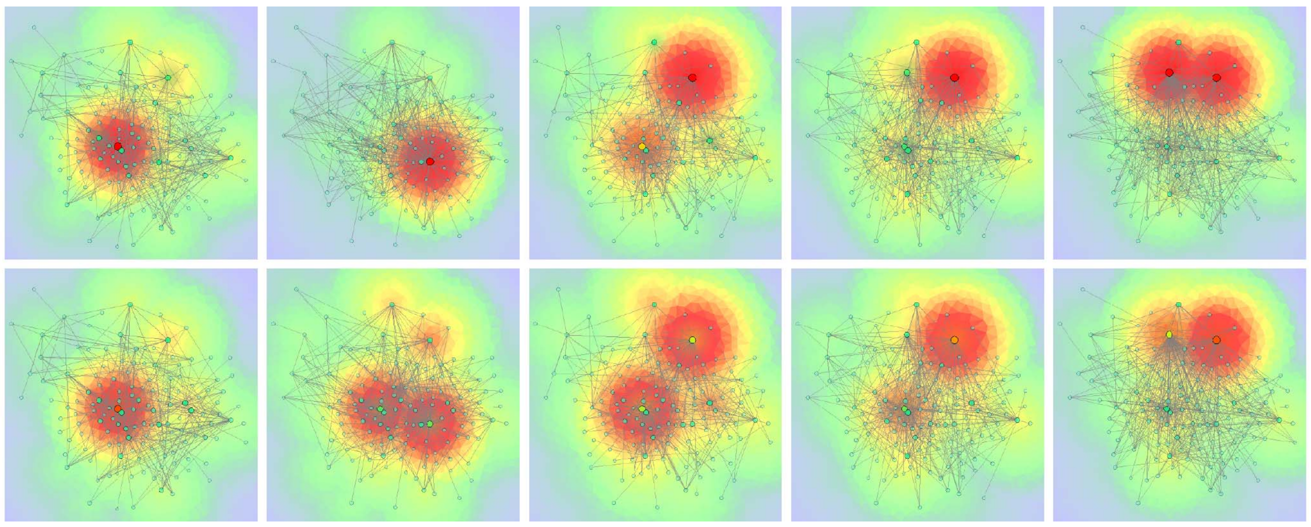


Fig. 7. Top row, left to right: triangle meshes showing the significance of the time-varying graph for five consecutive time steps without blending node importance over time. Bottom row, left to right: the corresponding triangle meshes with blending. The size of the time windows is 10 in this example.

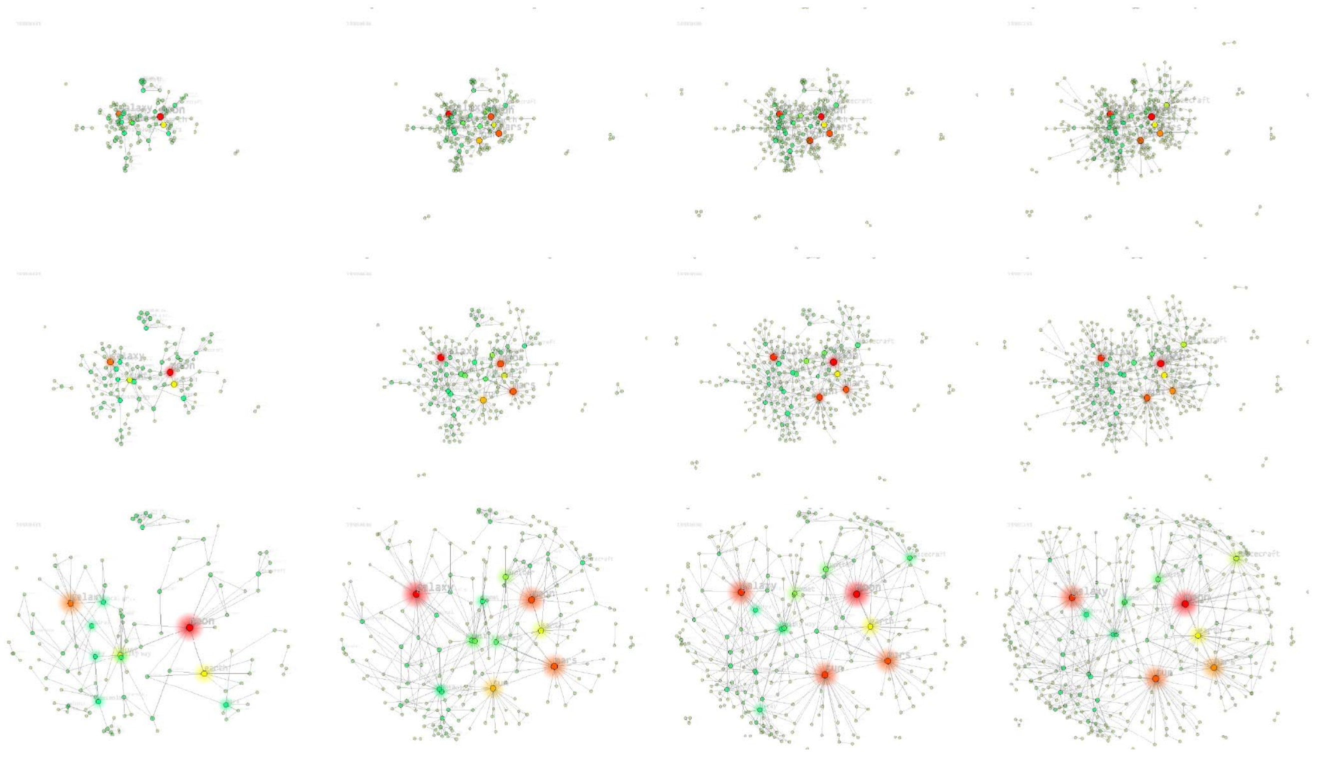


Fig. 8. Left to right: four selected time steps of the astronomy tag data set. Top to bottom: the initial graph layouts extracted from the super graph, the adjusted layout only considering aesthetic balance and temporal coherence, and the final layout after mesh deformation, respectively.

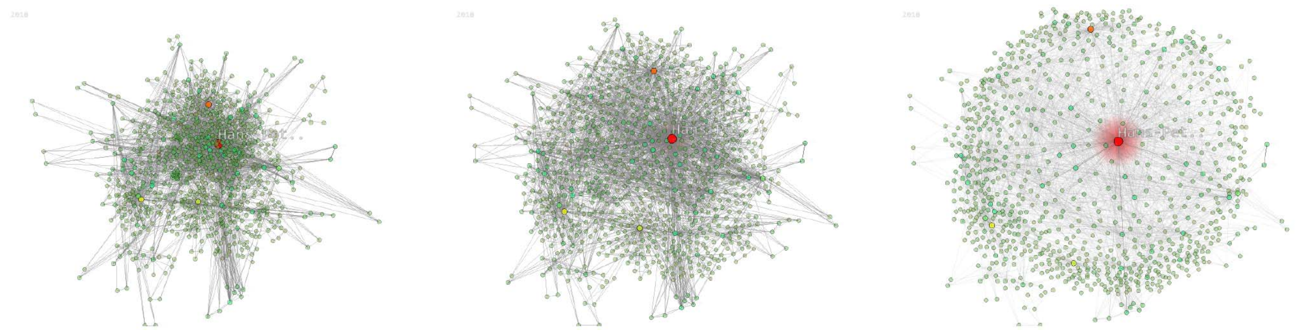


Fig. 9. The graph layouts for one selected time step of the DBLP coauthorship data set. Left: the initial graph layouts extracted from the super graph. Middle: the adjusted layout only considering aesthetic balance and temporal coherence. Right: the final layout after mesh deformation.

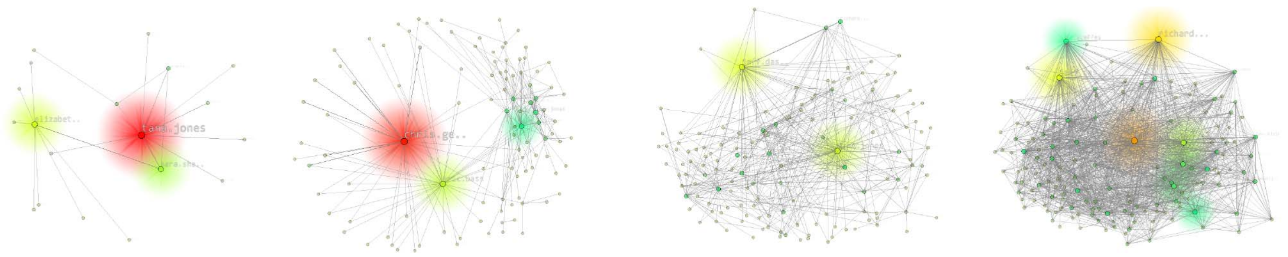


Fig. 10. Left to right: the final layout of four selected time steps of the Enron e-mail data set.

$1/3$), but the user can always customize the degree of mental map preservation for the most effective viewing.

5 Discussion

Our work on time-varying graph visualization is inspired by Wang et al. [29], [30], [31]. While they solved the F+C problems for static polygon and volume data, a direct application of these methods [30], [31] by adding spatio-temporal coherence terms [29] to time-varying graphs does not lead to smooth F+C visualization. This is because unlike video frames, which tend to change more smoothly, a time-varying graph can experience more abrupt changes in the location, size, and connectivity 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 nodes, we need

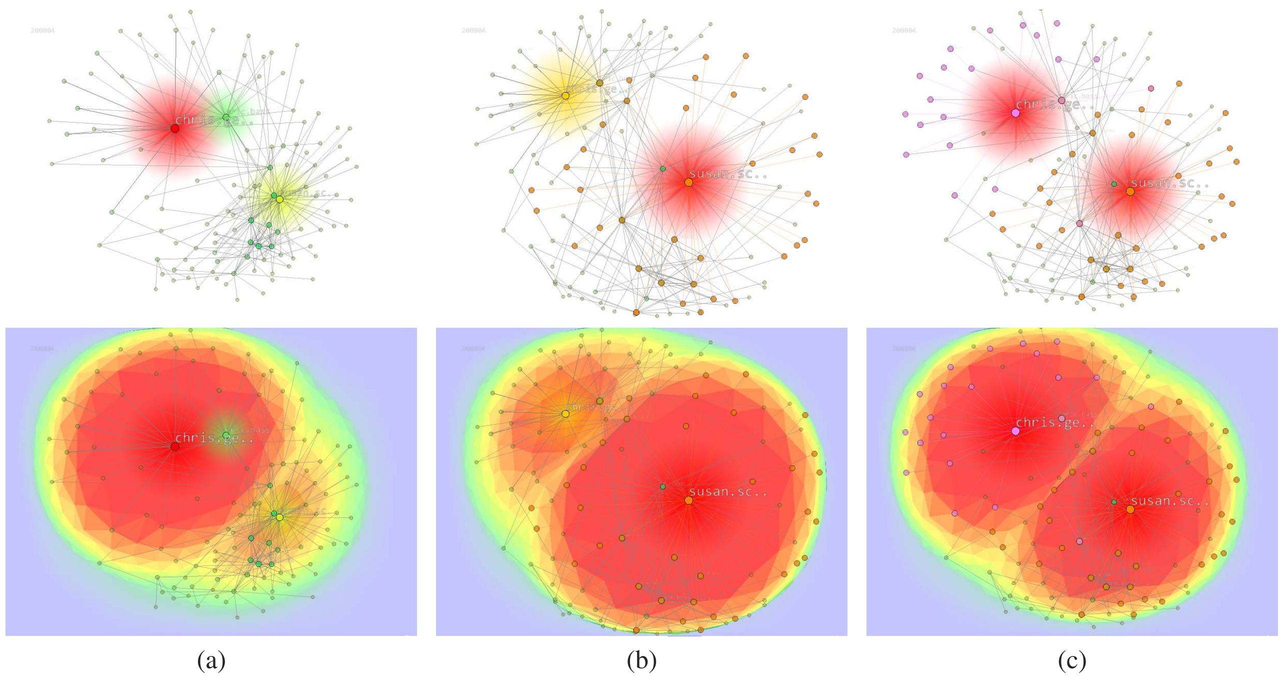


Fig. 11. Multi-F+C Visualization. (a) The initial graph and its corresponding triangle mesh. (b) The result with a single focus. (c) The result with two foci.

special treatments beyond what is presented in [30], [31] to achieve satisfactory results.

Our work addresses two major limitations in the original super graph algorithms [5], [6]. First, while the super graph algorithms preserve the mental map using the global layout for a given sequence of graphs, they did so at the cost of certain aesthetic criteria. Diehl and Görg [5] solved this problem by compromising aesthetic quality and dynamic stability, which 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 as nodes in the focus change in size and location. In this case, it is very difficult to maintain spatiotemporal coherence of the graphs.

We combine the super graph with the deformation model to generate smooth F+C visualization for time-varying graphs. The advantage of this 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 maintain high aesthetic quality using spatiotemporal energy terms in the deformation and solve the problem of Diehl et al. [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. The second advantage is that our energy minimization approach can maintain spatiotemporal coherence for nodes of various weights while our deformation model achieves stable F+C viewing. 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 a single optimization framework.

6 Conclusions and Future Work

We have presented a new solution to visualize time-varying graphs that allows users to generate customized layouts and animations via simple interaction. We achieve this by utilizing the ideas of the super graph and graph triangulation to produce a smooth and coherent visualization with multi-F+C capability. By transforming the graph layout problem to a constrained optimization problem for mesh deformation, we improve the layout directly extracted from the super graph while highlighting nodes and their surrounding regions of interest. Through adjusting the importance of nodes, the users can dynamically change the significance distribution in the graph and observe the new layout. They can also specify different nodes in the focus at a certain time step and rearrange the graph layout effectively via GPU acceleration. Importance-driven time budget allocation produces an animation with an emphasis on important time steps, thus facilitating detailed analysis of the graph.

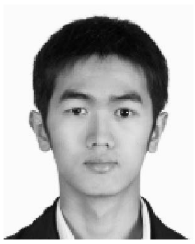
In this paper, we utilize the definitions of degree centrality and authority from graph theory to determine node importance. There exist other ways of defining degree centrality such as betweenness centrality, closeness centrality, and eigenvector centrality [32]. All these can be utilized and incorporated 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 a mesh, we cannot change the relative node positions in the initial graph layouts. In some extreme cases, the initial layouts may not exhibit spatiotemporal coherence at all and the deformation could lead to undesired layouts by breaking and flipping node relationships 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 time and the results could be very complex. In the future, we would like to improve our algorithm by taking a multilevel approach to prioritize nodes for time-varying graph visualization. This would reduce the complexity of graph and speed up the computation, providing a more efficient way to visualize large time-varying graphs.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their insightful comments. This work was supported in part by the National Science Council (contracts NSC-99-2221-E-006-066-MY3, NSC-100-2628-E-006-031-MY3, NSC-100-2221-E-006-188-MY3), Taiwan, and was supported in part by Michigan Technological University startup fund and the US National Science Foundation (NSF) through grant IIS-1017935.

REFERENCES

[1] U. Brandes, "Drawing on Physical Analogies," *Drawing Graphs: Methods and Models,* M. Kaufmann and D. Wagner, eds., pp. 71-86, Springer-Verlag, 2001.
[2] U. Brandes and D. Wagner, "A Bayesian Paradigm for Dynamic Graph Layout," *Proc. Int'l Symp. Graph Drawing,* pp. 236-247, 1997.
[3] J. Branke, "Dynamic Graph Drawing," *Drawing Graphs: Methods and Models,* M. Kaufmann and D. Wagner, eds., pp. 228-246, Springer-Verlag, 2001.
[4] L. Buatois, G. Caumon, and B. Lévy, "Concurrent Number Cruncher: A GPU Implementation of a General Sparse Linear Solver," *Int'l J. Parallel, Emergent and Distributed Systems,* vol. 24, no. 3, pp. 205-223, 2009.
[5] S. Diehl and C. Görg, "Graphs, They Are Changing," *Proc. Int'l Symp. Graph Drawing,* pp. 23-30, 2002.
[6] S. Diehl, C. Görg, and A. Kerren, "Preserving the Mental Map Using Foresighted Layout," *Proc. Eurographics - IEEE TCVG Symp. Visualization,* pp. 175-184, 2001.
[7] P. Eades, "A Heuristic for Graph Drawing," *Congressus Numerantium,* vol. 42, pp. 149-160, 1984.
[8] Y. Frishman and A. Tal, "Dynamic Drawing of Clustered Graphs," *Proc. IEEE Symp. Information Visualization,* pp. 191-198, 2004.
[9] Y. Frishman and A. Tal, "Online Dynamic Graph Drawing," *Proc. Eurographics - IEEE VGTC Symp. Visualization,* pp. 75-82, 2007.
[10] T.M.J. Fruchterman and E.M. Reingold, "Graph Drawing by Force-Directed Placement," *Software—Practice and Experience,* vol. 21, no. 11, pp. 1129-1164, 1991.
[11] G.W. Furnas, "Generalized Fisheye Views," *ACM SIGCHI Bull.,* vol. 17, no. 4, pp. 16-23, 1986.
[12] G.W. Furnas, "A Fisheye Follow-Up: Further Reflections on Focus+Context," *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems,* pp. 999-1008, 2006.
[13] E. Gansner, Y. Koren, and S. North, "Topological Fisheye Views for Visualizing Large Graphs," *Proc. IEEE Symp. Information Visualization,* pp. 175-182, 2004.
[14] E.R. Gansner and Y. Hu, "Efficient, Proximity-Preserving Node Overlap Removal," *J. Graph Algorithms and Applications,* vol. 14, no. 1, pp. 53-74, 2010.
[15] E.R. Gansner, Y. Koren, and S. North, "Graph Drawing by Stress Majorization," *Proc. Int'l Symp. Graph Drawing,* pp. 239-250, 2005.
[16] C. Görg, P. Birke, M. Pohl, and S. Diehl, "Dynamic Graph Drawing of Sequences of Orthogonal and Hierarchical Graphs," *Proc. Int'l Symp. Graph Drawing,* pp. 228-238, 2005.
[17] D. Harel and Y. Koren, "Graph Drawing by High-Dimensional Embedding," *Proc. Int'l Symp. Graph Drawing,* pp. 299-345, 2002.
[18] T. Kamada and S. Kawai, "An Algorithm for Drawing General Undirected Graphs," *Information Processing Letters,* vol. 31, no. 1, pp. 7-15, 1989.
[19] G. Kumar and M. Garland, "Visual Exploration of Complex Time-Varying Graphs," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 5, pp. 805-812, Sept. 2006.
[20] K. Misue, P. Eades, W. Lai, and K. Sugiyama, "Layout Adjustment and the Mental Map," *J. Visual Languages and Computing,* vol. 6, no. 2, pp. 183-210, 1995.
[21] A. Noack, "An Energy Model for Visual Graph Clustering," *Proc. Int'l Symp. Graph Drawing,* pp. 425-436, 2004.
[22] S.C. North, "Incremental Layout in DynaDAG," *Proc. Int'l Symp. Graph Drawing,* pp. 409-418, 1996.
[23] H.C. Purchase, E. Hoggan, and C. Görg, "How Important Is the "Mental Map"? - An Empirical Investigation of a Dynamic Graph Layout Algorithm," *Proc. Int'l Symp. Graph Drawing,* pp. 184-195, 2006.
[24] H.C. Purchase and A. Samra, "Extremes Are Better: Investigating Mental Map Preservation in Dynamic Graphs," *Proc. Int'l Symp. Graph Drawing,* pp. 60-73, 2008.
[25] M. Sarkar and M.H. Brown, "Graphical Fisheye Views of Graphs," *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems,* pp. 83-91, 1992.
[26] 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," *Proc. ACM Symp. User Interface Software and Technology,* pp. 81-91, 1993.
[27] J.R. Shewchuk, "Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator," *Proc. ACM Workshop Applied Computational Geometry,* pp. 203-222, 1996.
[28] C. Wang, H. Yu, and K.-L. Ma, "Importance-Driven Time-Varying Data Visualization," *IEEE Trans. Visualization and Computer Graphics,* vol. 14, no. 6, pp. 1547-1554, Nov./Dec. 2008.
[29] Y.-S. Wang, H. Fu, O. Sorkine, T.-Y. Lee, and H.-P. Seidel, "Motion-Aware Temporal Coherence for Video Resizing," *ACM Trans. Graphics,* vol. 28, no. 5, p. 127, 2009.
[30] Y.-S. Wang, T.-Y. Lee, and C.-L. Tai, "Focus+Context Visualization with Distortion Minimization," *IEEE Trans. Visualization and Computer Graphics,* vol. 14, no. 6, pp. 1731-1738, Nov. 2008.
[31] Y.-S. Wang, C. Wang, T.-Y. Lee, and K.-L. Ma, "Feature-Preserving Volume Data Reduction and Focus+Context Visualization," *IEEE Trans. Visualization and Computer Graphics,* vol. 17, no. 2, pp. 171-181, Feb. 2011.
[32] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications.* Cambridge Univ. Press, 1994.



Kun-Chuan Feng received the BS degree in computer science and information engineering from National Chung Cheng University, Taiwan, in 2008, and the MS degree from the Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan, in 2010. His research interests include computer graphics, image resizing, and visualization.

Chaoli Wang received the BE and ME degrees in computer science from Fuzhou University, China, in 1998 and 2001, respectively, and the PhD degree in computer and information science from The Ohio State University in 2006. He is an assistant professor of computer science at Michigan Technological University. His research focuses on large-scale data analysis and visualization, high-performance computing, and user interfaces and interaction. From 2007 to 2009, he was a postdoctoral researcher at the University of California, Davis. He is a member of the IEEE.

Han-Wei Shen received the BS degree from the Department of Computer Science and Information Engineering, National Taiwan University, in 1988, the MS degree in computer science from the State University of New York at Stony Brook in 1992, and the PhD degree in computer science from the University of Utah in 1998. He is an associate professor at The Ohio State University. From 1996 to 1999, he was a research scientist at NASA Ames Research Center in Mountain View, California. His primary research interests are scientific visualization and computer graphics. He is a winner of the US National Science Foundation (NSF) CAREER Award and the US Department of Energy (DOE) Early Career Principal Investigator Award. He also won the Outstanding Teaching Award twice in the Department of Computer Science and Engineering at The Ohio State University.

Tong-Yee Lee received the PhD degree in computer engineering from Washington State University, Pullman, in May 1995. He is currently a distinguished professor in the Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan, ROC. He leads the Computer Graphics Group, Visual System Laboratory, National Cheng Kung University (http://graphics.csie.ncku.edu.tw/). His current research interests include computer graphics, nonphotorealistic rendering, medical visualization, virtual reality, and media resizing. He also serves on the editorial boards of the *IEEE Transactions on Information Technology in Biomedicine*, the *Visual Computer and the Computers and Graphics Journal*. He served as a member of the international program committees of several conferences including the IEEE Visualization, the Pacific Graphics, the IEEE Pacific Visualization Symposium, the IEEE Virtual Reality, the IEEE-EMBS International Conference on Information Technology and Applications in Biomedicine, and the International Conference on Artificial Reality and Telexistence. He is a senior member of the IEEE and a member of the ACM.