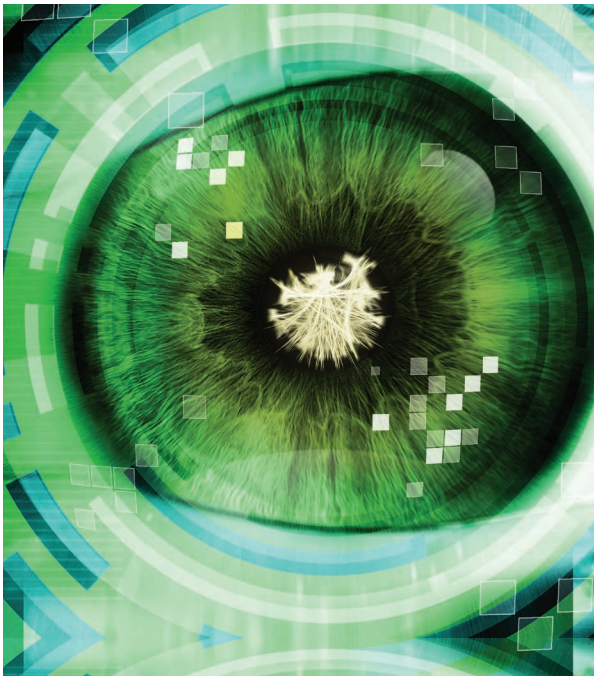

Large-Scale Graph Visualization and Analytics

Kwan-Liu Ma and Chris W. Muelder, *University of California, Davis*

Novel approaches to network visualization and analytics use sophisticated metrics that enable rich interactive network views and node grouping and filtering. A survey of graph layout and simplification methods reveals considerable progress in these new directions.

Relational data is one of the primary classes of information and appears in a wide array of disciplines, ranging from sociology and biology to engineering and computer science. Unlike spatial, *n*-dimensional, or text data, relational data consists of a set of entities and a network of relationships among them. Some networks represent abstract relationships, such as author influence or friendship; others represent physical networks, such as power distribution or routers.

With the growing popularity of mainstream network applications, the ability to efficiently analyze complex data collections has become critical. Wikipedia has millions of articles that form a network through cross-references. Facebook connects more than a billion users in an incredibly complex structure of friends, group invitations, games, advertising, video chats, and so on. These and similar networks continue to expand and evolve daily.

Using simple statistics to reason about the dynamics of such complex networks is not generally effective or practical. Rather, analysts are turning to visualization—not just the passive process of producing images from numbers, but highly interactive methods that combine visual representations with network analytics to greatly enhance the ability to understand and characterize networks. Such analysis can yield important insights. Social network analysis, for example, can reveal patterns about groups of friends or popularity, and the analysis of a power distribution network can indicate key points for infrastructure improvements.

Graph drawing, which began in the 1960s, is a field of research dedicated to visualizing a network's structure. One of the most common and intuitive representations is the node-link diagram, in which nodes represent actors and the links between nodes represent the actors' interrelationships. Although this method is relatively straightforward and practical for visualizing small networks, it can be overwhelming for large, complex networks.

Time-varying networks, such as Facebook, impose additional challenges. A social network grows with each new friendship or alliance and shrinks when friendships grow distant or break apart. Because each node addition or deletion can affect larger-scale patterns, such as clusters, finding and understanding small changes can provide insights into the entire network's evolution.

0018-9162/13/$31.00 © 2013 IEEE Published by the IEEE Computer Society

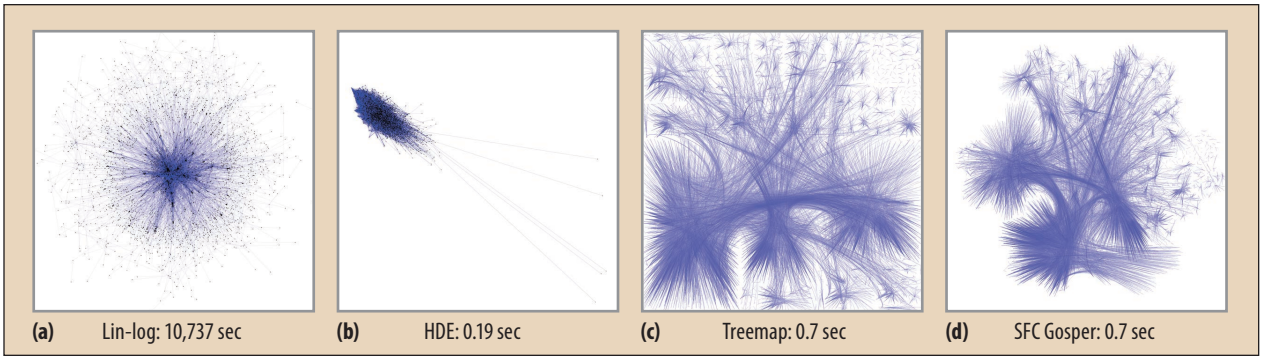

Figure 1. Visualization approaches to lay out a static graph of website hyperlinks in a search for the word "California." Both (a) a traditional force-directed layout method such as lin-log and (b) an algebraic approach such as high-dimensional embedding (HDE) yield an unintelligible hairball—a tangled mess of lines. Clustering-based layout algorithms, such as the (c) treemap and (d) space-filling curve (SFC) layouts, offer better representations of higher-level network structures. Timing is based on the use of a Mac Pro with a 2.66-GHz Intel Xeon CPU.

Graph visualization and analysis are ripe for research innovation to address the escalating scale and complexity of data and information systems. New methods must address all aspects of network representation, from the fundamental problem of laying out a large graph to graph analytics and simplification for dynamic graphs. To work toward this goal, we have been identifying open problems and developing corresponding solutions. These efforts show the promise of visualization to support new forms of exploratory network data analysis.

GRAPH LAYOUT

One challenge in node-link diagrams is how to efficiently provide a node placement or layout that will yield a meaningful graph visualization. For simple structures, the system needs only a set of aesthetic choices to provide a useful graph—sometimes even a hand-drawn visualization could suffice. But for large, complex structures, effective layouts are harder to create, which motivates continual interest in graph layout algorithms as an integral part of visualizing complex networks. Although most traditional work involves developing more efficient layout methods for static graphs, more recent efforts have also focused on finding effective ways to generate dynamic graphs of time-varying networks.

Static graphs

Algorithms for laying out static graphs vary considerably. Figure 1a shows the results of a force-directed layout method, which arranges graphs by iteratively refining the positions of vertices to incrementally reduce an energy function. The energy function varies between different force-directed algorithms, but it is generally a function of the distances between nodes and the weights of the edges between them. Although force-directed layouts are generally aesthetic, their algorithms do not scale well to large or dense graphs. Lin-log[1] is an example of a force-directed layout method.

More efficient layout algorithms use a multiscale approach, such as FM (fast multipole multilevel method) and GRIP (graphic drawing with intelligent placement).[2,3] Both of these algorithms begin by laying out a small approximation of the graph, and then progressively laying out finer approximations until they complete the entire original graph. Multiscale algorithms tend to need far fewer iterations than force-directed approaches and thus have a vastly superior performance.

Faster than either force-directed or multiscale layout methods, algebraic methods use linear algebra techniques instead of force calculations to calculate the layout directly.[2,3] Algebraic methods tend to work well on regular, grid-like networks, but can fail to produce good layouts for denser real-world networks. Examples of algebraic layout methods include ACE (algebraic multigrid computation of eigenvectors), HDE (high-dimensional embedding), and Maxent. Figure 1b shows a graph laid out with the HDE method.

Other layouts gain efficiency by clustering the graph in a preprocessing step and then using the clustering to define the layout itself. This two-stage process allows the layout to adapt to changes extremely quickly, making it highly suitable for interactive visualizations. Figures 1c and 1d depict layouts using two such layout methods. The treemap layout method[4] uses a hierarchical space decomposition to map a hierarchical clustering to the screen: the root of the clustering's tree takes up the whole screen, and each branch recursively subdivides the screen according to the clustering until finally each leaf is allotted its own region. Each leaf represents a node, so placing each node in its corresponding region defines the layout. The space-filling curve (SFC) layout method[5] uses the clustering to order nodes in one dimension and then applies a recursively defined fractal curve such as the Hilbert or Gosper curve to map them to the screen.

The treemap method maps clusters directly to the screen, which guarantees that each node has equal

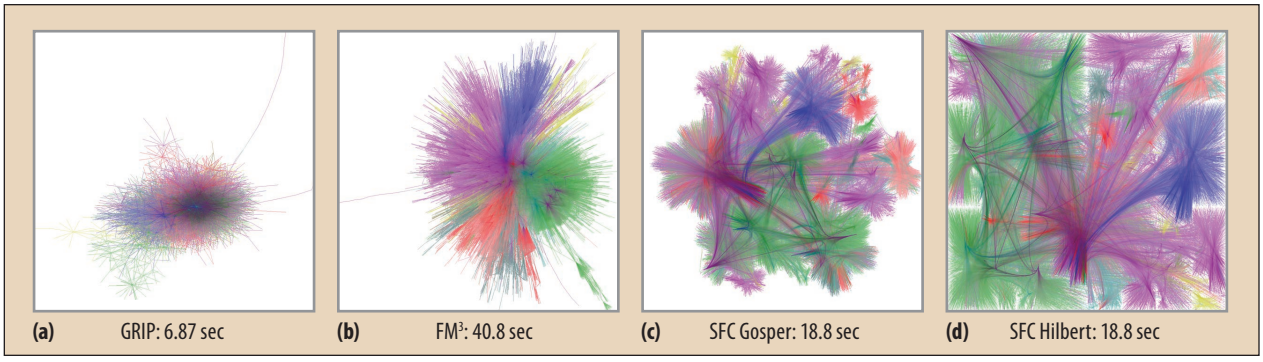

Figure 2. Layouts of the Internet's physical connectivity at the autonomous system level with 41,928 nodes and 218,080 edges, colored by continent. Although force-directed approaches such as (a) graphic drawing with intelligent placement (GRIP) and (b) the fast multipole multilevel method (FM) help arrange nodes according to connectivity and implicitly by country, they yield unreadable hairballs. (c,d) Clustering-based approaches, such as the SFC algorithm, clearly define groups of high interconnectivity, and edge bundling elucidates intercluster relationships. Times are from a Linux machine with a 3.20-GHz Intel Core i7 960 CPU.

screen space and that clusters are clearly separated. However, it does not guarantee aspect ratios, which can yield unreadably thin clusters. The SFC method, on the other hand, maps clusters onto segments of a fractal curve, which does guarantee good aspect ratios. Both the treemap and SFC approaches meet common aesthetic criteria, such as cluster collocation and short average edge length. Because of their clustering basis, they work best on networks that form strong clusters, such as biological and social networks and Web hyperlinks. They do not work well on regular, mesh-like networks. However, real-world networks are scale-free and rarely form regular meshes.

As Figure 1 shows, the times and resulting graphs are dramatically different across layout methods. The graph is modestly sized with only 6,107 nodes and 15,160 edges, yet lin-log took almost three hours to generate a layout, and the result is still a hairball. Even though HDE generated a layout in subseconds, the result was even less useful than lin-log's. In sharp contrast, both the treemap and the SFC layouts depict a clearer number and relative size of clusters, use screen space more efficiently, and elucidate intercluster relationships with routed edges.

Many visual analytics tasks for decision making require a rapid overview visualization, and in our results, only the clustering-based layout methods, such as SFC, provided rapid, insightful results—particularly with larger networks. As Figure 2 shows, relative to the GRIP and FM^3 force-directed methods, the SFC methods clearly use the screen space more efficiently than the other methods, and the clusters are more clearly distinct.

Dynamic graphs

Researchers have extensively studied the problem of visualizing static networks, but work on dynamic network visualization is still in its infancy. Any layout method to create a node-link diagram of a dynamic graph must account for both the graph topology and stability between time steps. The result is a tradeoff between layout quality and stability: a perfectly stable layout would sacrifice layout quality, and naively calculating ideal layouts would not offer stability. Many researchers refer to this stability problem as preserving the mental map.

Some experiments have examined the usefulness of preserving the mental map in dynamic graph visualizations. In one study,[6] the results were surprising because the most effective visualizations were extreme cases—with very low or high mental map preservation—rather than with medium preservation.

A common method for visualizing dynamic graphs is to animate the transitions between time steps. The result is a dynamic visualization, in which nodes appear, disappear, and move to produce a readable layout for each time step. An alternative is to use small multiples to statically place multiple time steps next to each other.[7] This approach makes it easier to compare distant time steps but limits the area devoted to each time step, thereby reducing each graph's legibility. An empirical study[8] has compared the advantages and drawbacks of using animation or small multiples.

Another approach uses time as a dimension in the static visualization of dynamic graphs.[9] The algorithm orders vertices and positions them on several vertical parallel lines, using directed edges to connect the vertices from left to right. The results display each time step's graph between two consecutive vertical axes. Another method uses a geographical metaphor to visualize clustered dynamic graphs.[10] However, this method uses a priori global clustering over time and does not handle cluster evolution.

Animation at a very large scale can be overwhelming, so it is necessary not only to minimize motion but also to stabilize animation. Recent work has produced two clustering-based layouts both of which achieve scalability far beyond what other animation methods have demonstrated.

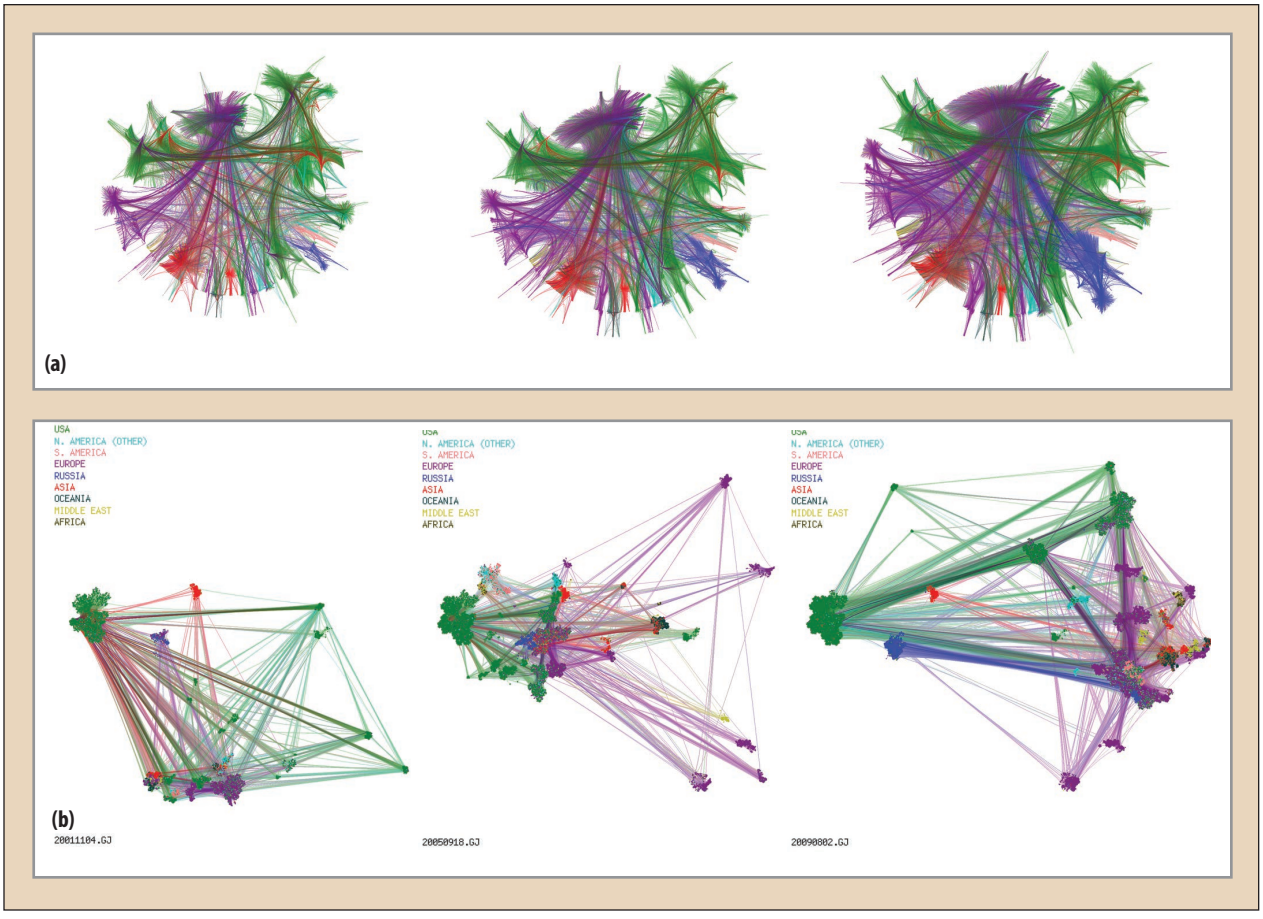

Figure 3. Dynamic graph layout results for visualizing evolving Internet connectivity. (a) The incremental-clustering-based approach uses space efficiently. Motion is slow, smooth, and affine, and so is easy to follow, but quality degrades over time to ensure stable animation. (b) The global-clustering-based approach meets layout criteria—balanced quality and stability with nodes largely remaining stable, but clusters are compacted. The images are freeze frames of three time steps. Videos of the complete dataset of 400 time steps are available at http://vis.cs.ucdavis.edu/Videos/Computer2013.

An incremental-clustering-based layout[11] method starts by laying out a single time step, then incrementally modifies the underlying clustering over time, which updates the layout algorithm accordingly. The resulting visualization starts with the rapid generation of an ideal layout, but over time trades off quality to gain stability.

Figure 3a shows the results for three time steps in Internet connectivity evolution. Nodes are not completely stationary, but node clusters move slowly and affinely as a coherent group.

A global-clustering-based layout method uses the whole time range and aims to optimally satisfy both the layout quality for each time step and the layout's stability over time. One method extends SFC to create such a layout.[12] Rather than trying to find a single global clustering, this method clusters each time step independently and then associates clusters pairwise between neighboring time steps to track the clusters over time. It then orders both clusters and nodes globally to minimize energy functions. The ordered clusters and nodes define both a time-varying layout by mapping individual time steps to the screen with SFC and a static timeline by directly plotting the ordering against time. Because the method precalculates the entire ordering, analysts can use this timeline to interactively explore the graph not only sequentially but also by navigating between arbitrary time steps.

Figure 3b shows the results for the same three time steps portrayed in Figure 3a. The layout minimizes intracluster edge lengths but leaves space for nodes that are not in the given time steps. Because it clusters each time step independently, global optimization guarantees good layout quality properties at all times. The ordering method also minimizes node motion, which aids in mental map preservation. In this way, the approach balances quality and stability.

Although both approaches have strong advantages, they also have drawbacks. The incremental approach does not guarantee that the layout is ideal for time steps that are distant from the initially clustered step. Globally optimized layouts have compacted clusters, because they

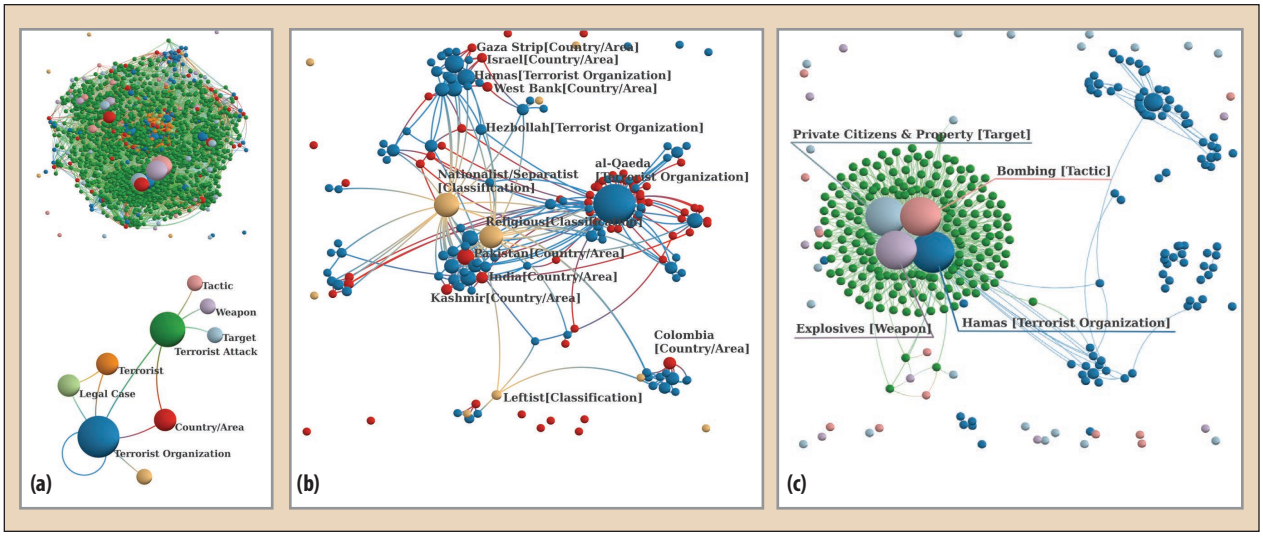

Figure 4. Visualization of a heterogeneous terrorist network with semantic filtering. (a) At the top is a naive force-directed layout of the entire terrorist network (2,374 nodes and 8,767 links), which is too dense and cluttered to be useful. At the bottom is a network ontology—a graph of the kinds of nodes and connections that occur in the full network—which can filter the visualization and reveal useful structures. (b) Visualizing the terrorist organization (blue), locations (red), and classification (tan) reveals which organizations are active in which regions. (c) Visualizing organizations, weapons, tactics, and targets reveals attack behaviors, such as Hamas's preference to bomb civilians.

reserve empty space for past and future clusters. The global optimization's main limitation is computational complexity. Cluster and node ordering in particular take a large amount of computation—overnight for the dataset in Figure 3. Current efforts aim to alleviate these drawbacks by combining both approaches by developing incremental clustering and more efficient ordering algorithms.

GRAPH SIMPLIFICATION

Even with optimizations, directly laying out an entire graph with tens of millions of nodes or edges does not always lend itself well to detailed analysis. A more practical and cost-effective method is to extract subgraphs from the overall graph or to simplify the graph according to certain metrics. The resulting smaller graphs are then easier to display and analyze quickly.

Semantic abstraction

Structural visualizations are often less effective for large, dense graphs. Node-link diagrams can become rapidly cluttered, and algorithmic layouts often result in hairballs. When a network contains heterogeneous elements and if the relationship of interest is between a small subset of element types, showing only the relevant connections can be more intuitive than a full layout. An *ontology*—a graph whose nodes represent node types and whose links represent association types—is useful in extracting such explicit relationships. Ontologies can be based on existing explicit relationships or connections or on inferences from cross-referencing external sources—for example, a database of node types might be the basis for classifying edges according to the types of their associated nodes (*A* to *A*, *A* to *B*, *B* to *B*, and so on).

Ontologies represent the types of actors and relations in a network. OntoVis uses a high-level graphical representation of an ontology to enhance a typical node-link diagram,[13] providing an interface to control the ontological nature of heterogeneous networks and enable user-directed filtering to isolate subnetworks of interest. Figure 4 shows an OntoVis application to visualize a terrorist network.[13]

Another type of visualization that relies on the meaning of actors and links is based on *semantic substrates*[14]—spatial network layouts in which node position depends on the values of a given node attribute. The system can display multiple semantic substrates simultaneously and connect them according to the links between them. Because node placement follows attribute values, not the network's structure, the semantic substrate metaphor is a better fit for semantic queries.

Centrality sensitivity

A popular statistical metric for social and other scale-free networks is *centrality*, which quantifies each node's importance to network's structure. Central nodes are often important hubs through which social interaction happens and are good indicators of individual nodes' and clusters' relative popularity. Node centrality is generally a function of structural variables, such as degree or, more generally, a function of the network's adjacency matrix.

Centrality sensitivity[15] measures how the change of each node's importance influences other nodes' importance. As

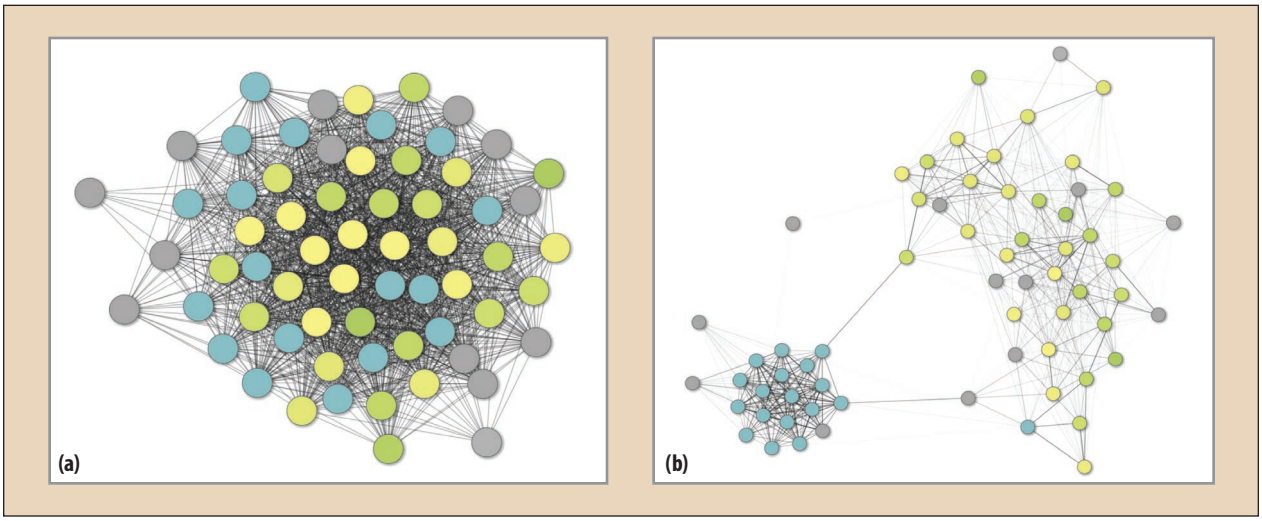

Figure 5. Sensitivity-based visualization of proximity network of mobile phone users (the MIT Reality Data). (a) A force-directed layout does not convey groups, except via semantic attributes (blue for Sloan school, green for MediaLab members). (b) Filtering according to only positive centrality sensitivities and weighing edges based on sensitivity magnitude derives a clear separation between two user groups, and also reveals that the Sloan members are much more tightly connected.

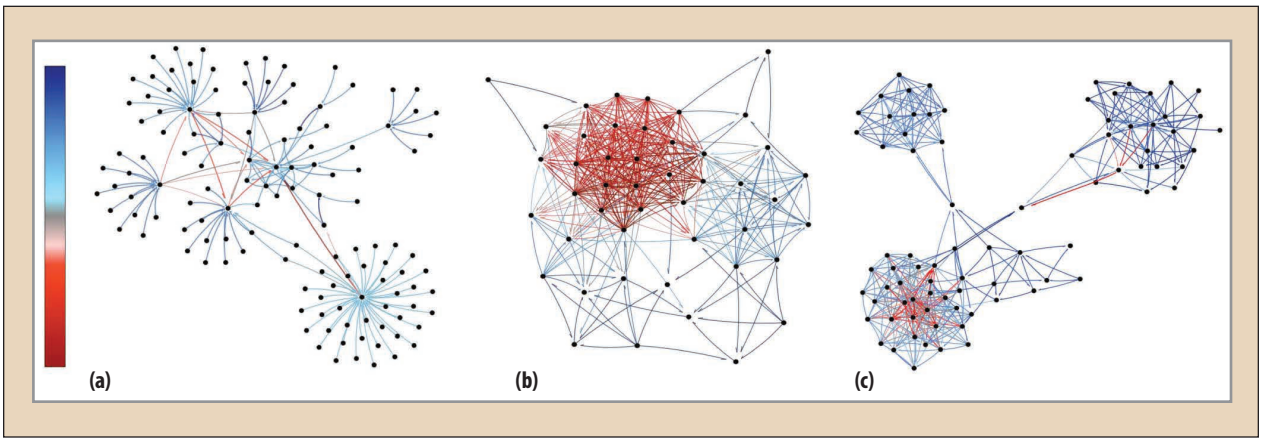

Figure 6. Core network visualization using centrality sensitivity. Color encoding sensitivities helps identify interaction types. Blue and red edges denote positive and negative sensitivity, respectively. (a) A graph of a Friendster social network forms a connected collection of star-shaped networks, in which links between core nodes exhibit negative sensitivity. (b) An astrophysics co-citation network forms a dense competitive core, with the periphery exhibiting more collaborative ties. (c) A network from Del.icio.us exhibits both behaviors, with a core network linking tightly connected groups.

such, sensitivities are good indicators of how centrality is distributed and how the system is likely to propagate network changes.

Analytical or approximated solutions compute sensitivities for different centrality metrics. Such sensitivity analysis can be helpful in interpreting visualization because it is adept at finding hidden relationships in a network that could otherwise be overlooked in direct visual representation.

Centrality sensitivity is useful as a metric for simplifying complex networks and supporting visual reasoning. The simplified network retains key structural properties while maintaining a trustworthy, readable visualization. Figure 5 shows the use of the sign and magnitude of sensitivities to characterize clusters in a mobile phone user network. Filtering according to centrality sensitivity separates users into two groups and reveals that users in one group appear to maintain a tighter bond than in the other.

Centrality sensitivity also reveals hard-to-find collaborative or competitive relations. In the uncertainty analysis of social networks, it helps analysts gain insight into the robustness of key network metrics. Figure 6 shows several social networks colored according to centrality sensitivity, with red for negative and blue for positive sensitivity. Figure 6a shows the Friendster network as a core of central nodes, many of which have a connected star pattern of isolated nodes. These peripheral nodes have a collaborative relationship (positive sensitivity) with the

core nodes because they depend on each other, but the ties between core nodes are more competitive (negative sensitivity), since an increase in any one node's importance would reduce the importance of the others.

The co-citation network from ArXiv, in Figure 6b, contains a highly competitive clique—any increase in a cluster member will negatively impact the importance of the others. The network is in a somewhat unstable state, since each cluster member has a roughly equal chance of becoming the most important node.

In Figure 6c, the Del.icio.us network (a website-tagging social networking site) exhibits collaborative clusters (top) as well as a competitive cluster (bottom), with a competitive skeleton network between them.

When it is infeasible to fully comprehend the totality of a large network, particularly a dynamic network for time-critical applications, sampling is a common strategy.[16] However, random sampling strategies and their possible biases could lead to uncertain and biased samples. Sensitivity-based sampling could improve the results of statistical sampling by both guiding the sampling process and giving analysts a sense of the results' confidence level.

Finally, although most network analysis approaches we have described are top-down, visual analytics tasks such as visual recommendation can benefit from a bottom-up approach to network exploration, starting with a node or a tiny subset of nodes and expanding the region according to explicit connections to that node and those derived by metrics, and another work has shown that sensitivity-based approaches can be applied to this process.[17]

The explosive growth in size and complexity of real-world networks has overwhelmed conventional visualization and analysis methods. Although new approaches to graph layout, simplification, and analysis can make the analysis of complex networks more tractable, the resulting subgraphs could still be too large for interactive visualization. The convergence of analysis tools such as centrality and clustering analysis and interactive visualization have led to powerful visual analytics solutions.

The large scale of many dynamic networks remains a formidable hurdle. More effective time representations—beyond animations and time sliders—will provide deeper insights into how complex networks form and evolve. Network modeling and clustering are fundamental to addressing this challenge. Research should focus on developing more robust and efficient temporally aware clustering algorithms for dynamic graphs. Good clustering will produce layouts that meet general criteria, such as cluster colocation and short average edge length, as well as minimize node motion between time steps.

Unlike early systems that emphasize direct structural drawing, current visualization tools offer novel, interactive network views with the ability to filter and group nodes using sophisticated metrics. Semantic abstraction aids analysts in navigating large networks' complex spaces. Next-generation network analysis tools incorporating semantic and statistical views should help analysts better understand the structure of complex networks in different spaces.

Acknowledgments

This work is sponsored in part by the US National Science Foundation through grants CCF 0811422, CCF 0808896, and CCF 0634913. Thanks go to organizations/individuals who made publicly available datasets used in this work.

References

1. A. Noack, "An Energy Model for Visual Graph Clustering," LNCS 2912, Springer, 2004, pp. 425-436.
2. S. Hachul and M. Jünger, "An Experimental Comparison of Fast Algorithms for Drawing General Large Graphs," *Proc. 13th Int'l Symp. Graph Drawing* (GD 04), LNCS 3843, Springer, 2005, pp. 235-250.
3. E.R. Gansner, Y. Hu, and S. North, "A Maxent-Stress Model for Graph Layout," *Proc. IEEE Pacific Visualization Symp.* (PacificVis 12), IEEE, 2012, pp. 73-80.
4. C. Muelder and K.-L. Ma, "A Treemap-Based Method for Rapid Layout of Large Graphs," *Proc. IEEE Pacific Visualization Symp.* (PacificVis 08), IEEE, 2008, pp. 231-238.
5. C. Muelder and K.-L. Ma, "Rapid Graph Layout Using Space Filling Curves," *IEEE Trans. Visualization and Computer Graphics*, vol. 14, no. 6, 2008, pp. 1301-1308.
6. H. Purchase and A. Samra, "Extremes Are Better: Investigating Mental Map Preservation in Dynamic Graphs," *Proc. 5th Int'l Conf. Diagrammatic Representation and Inference* (Diagrams 08), LNCS 5223, Springer, 2008, pp. 60-73.
7. E.R. Tufte, *Envisioning Information*, Graphics Press, 1990.
8. D. Archambault, H.C. Purchase, and B. Pinaud, "Animation, Small Multiples, and the Effect of Mental Map Preservation in Dynamic Graphs," *IEEE Trans. Visualization and Computer Graphics*, vol. 17, no. 4, 2011, pp. 539-552.
9. M. Burch et al., "Parallel Edge Splatting for Scalable Dynamic Graph Visualization," *IEEE Trans. Visualization and Computer Graphics*, vol. 17, no. 12, 2011, pp. 2344-2353.
10. Y. Hu, S.G. Kobourov, and S. Veeramani, "Embedding, Clustering, and Coloring for Dynamic Maps," *Proc. IEEE Pacific Visualization Symp.* (PacificVis 12), IEEE, 2012, pp. 33-40.
11. C.W. Muelder, "Advanced Visualization Techniques for Abstract Graphs and Computer Networks," PhD dissertation, Dept. Computer Science, University of Calif., Davis, 2011.
12. A. Sallaberry, C.W. Muelder, and K.-L. Muelder, "Clustering, Visualizing, and Navigating for Large Dynamic Graphs," *Proc. 20th Int'l Symp. Graph Drawing* (GD 12), LNCS 7704, Springer, 2012, pp. 487-497.
13. Z. Shen, K.-L. Ma, and T. Rad-Eliassi, "Visual Analysis of Large Heterogeneous Social Networks by Semantic and Structural Abstraction," *IEEE Trans. Visualization and Computer Graphics*, vol. 12, no. 6, 2006, pp. 1427-1439.

14. A. Aris and B. Shneiderman, "Designing Semantic Substrates for Visual Network Exploration Information Visualization," *Information Visualization*, vol. 6, no. 4, 2007, pp. 281-300.
15. C.D. Correa, T. Crnovrsanin, and K.-L. Ma, "Visual Reasoning about Social Networks Using Centrality Sensitivity," *IEEE Trans. Visualization and Computer Graphics*, vol. 18, no. 1, 2012, pp. 106-120.
16. J. Leskovec and C. Faloutsos, "Sampling from Large Graphs," *Proc. 12th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining* (KDD 06), ACM, 2006, pp. 631-636.
17. T. Crnovrsanin et al., "Visual Recommendations for Network Navigation," *Computer Graphics Forum*, vol. 30, no. 3, 2011, pp. 1081-1090.

Kwan-Liu Ma *is a professor of computer science at the University of California, Davis, as well as leader of the university's VIDI research group and director of the UC Davis Center for Visualization. His research interests include visualization, computer graphics, and high-performance computing. Ma received a PhD in computer science from the University of Utah. He is an IEEE Fellow and serves on the editorial boards of* IEEE Computer Graphics and Applications, Journal of Computational Science and Discoveries, *and* Journal of Visualization. *Contact him at ma@cs.ucdavis.edu.*

Chris W. Muelder *is a postdoctoral researcher at the University of California, Davis. His research interests are primarily in information visualization and visual analytics, including topics such as large or dynamic network analysis, parallel hardware performance visualization, computer network security visualization, and GPU applications to visualization. Muelder received a PhD in computer science from UC Davis. He is a member of ACM. Contact him at muelder@cs.ucdavis.edu.*