Visualizing Network Relationships

Scott Murray

Abstract—The vast majority of network visualizations are based on simple graphs and are rendered with connecting lines that communicate only one binary value: network nodes are either connected to each other or not. Information about the nature of the connections is either not available or not represented in these kinds of visualizations. *Relationship Visualizer* is an application that takes directed graphs with multiple edges as input and renders those edges visually meaningful. By incorporating directionality as well as assigning each edge a quantitative value, a new method of visualizing multiple-edge graphs is introduced. The application accepts input in a simple data format and employs basic user interaction tools to enable custom renderings of data sets. This approach has potential for visualizing any directed graph data with multiple edges and quantitative values, such as phone records, emails, social networks, economic trade data, website links, and network traffic.

Index Terms—Graph drawing, directed graphs, multiple edges, interactive graph visualization, network visualization.

1 Introduction

The vast majority of network visualizations use connecting lines that communicate only one binary value: network nodes are either connected to each other or not. Information about the nature of the connection—its strength, frequency, or direction—is either not available or not represented in these kinds of visualizations.

Extensive research has contributed incremental improvements to methods for drawing simple graphs, resulting in more efficient renderings and intelligently clustered nodes and edges that reduce visual clutter. Little work, however, has addressed visualizing directed graphs with multiple edges, specifically those in which each edge has associated quantitative values, the focus of this paper.

Working with more robust edge data provides the opportunity for each edge to convey not just a connection, but the nature of that connection. Since directed graphs incorporate the directionality of each connection, there are opportunities to visualize the (im)balances between nodes—the two-way nature of the relationships, in other words. Also, when working with multiple edges, each edge can be assigned a quantitative value, which could be a measure of time, a priority ranking, or any other relative value. Potential data sets for directed graphs with multiple edges include: phone records, emails, social networks, economic trade data, website links, and network traffic. Data in simple graph form (single edges only) could not be used with this new method.

2 Related Work

The majority of recent research on graph drawing tends to focus on improving the visual readability of representations of simple graphs. Useful approaches include clustering related nodes [1] and clustering related edges [2, 3] to reduce visual clutter, and even duplicating nodes [4] in specific applications in order to make trends and patterns more identifiable by human eyes.

An exhaustive search of recent literature in the field failed to discover any visualization research using directed graphs with multiple edges that encoded the edges as anything other than lines. A significant paper introducing a new algorithm for generating flow maps was found [5], but even that uses lines for edges, with edge values encoded as thickness. The only visualization research that uses bidirectional encoding [6] also employs lines.

3 Approach

The goal of this research was to develop a software tool that could:

- usefully visualize directed graphs with multiple edges,
- usefully visualize additional quantitative values associated with each edge,
- employ motion as a means of encoding data and making the end visualization more clear,
- employ interactivity and configurability to enable users to adjust the visualization to best suit their purposes, and
- take a very simple data format as input, to ease adoption of the tool for a wide range of purposes.

3.1 Useful Edge Representations

In the world of graph drawing, simple lines with uniform weight are common, but reveal only that two members are connected—only one bit of information. Yet lines may carry more visual weight and even occupy a greater area (whether in pixels or ink) than the nodes to which they connect, resulting in a low Tuftean "data-ink ratio" [7]. New visualization methods with more dense (and useful) information resolutions are needed.

By visualizing not just binary connections, but properties of the relationships between members, we could perceive visually which relationships are balanced, lopsided, one-way, or reciprocal, and identify similarities and differences across relationships.

A relationship-centric network visualization would de-emphasize a network's members in favor of the relationships between them. The first key contribution of this research is to *segregate individual edges into discrete visual forms*. This approach runs contrary to the conventional rendering of edges as lines, but by representing each edge as its own visual object, we both drastically increase the data-ink ratio as well as open up new possibilities for encoding even more data.

3.2 Encoding Quantitative Values for Each Edge

Taking Tufte's advice, if the visual representation is boring, we should start searching for more interesting data [7]. Fortunately, there is an enormous amount of data in the world that can be structured in directed graph form and used with this tool. And now, with each edge as a discrete visual form, we can attach a quantitative value to each edge and represent that value in the visualization.

For example, it may be useful to visualize patterns of email traffic within a company network. Each user's email address could be treated as a node, and each email as a directional edge, from node A to node B. But a network administrator may also be interested in the raw data volume of the exchanges, so a quantitative value of the *length* of each email is appended to the edge. Then, in the final visualization, the visual forms representing long messages could

- *Scott Murray is with the Massachusetts College of Art and Design, E-Mail: scott.murray@massart.edu.*

appear larger than those representing short ones. Patterns may emerge around users' email usage, and high-traffic exchanges would be visible at a glance.

3.3 Employing Motion

Existing interactive graph visualizations use motion only to animate node placement and adjust edge lines accordingly. *Relationship Visualizer*, however, employs motion (instead of arrows, for example) to indicate edge direction, so related edges travel along a path from node A to B, while opposing edges travel from B to A. While inappropriate use of motion has potential to distract and visually overwhelm the user (especially when viewing many nodes and edges), this development is nonetheless a new approach and may be considered a more intuitive representation for many users.

3.4 Interactivity and Configurability

This approach asserts that there is no single ideal visualization for any given data set. A visualization's success is best judged by its users, and each user will have differing needs and interests in the final output. Therefore, *Relationship Visualizer* employs basic interactivity (such as selection and dragging of nodes) and user-configurable parameters (such as rate of motion, and arc width) in an effort to enable each user to actively explore his or her data set and render the tool's visual output in the form that will be most meaningful to that user.

3.5 Simplified Data Format

To encourage easy use with the widest possible range of data sets, the tool has been designed to accept input data in a very simple, accessible format (see section 4.1 below).

Processing, a free, open-source programming environment, was chosen as the development platform, in part due to its strengths in easily capturing and parsing data sets, but also to ensure accessibility of the tool to others after its release. Version 1.0 or newer is required to use *Relationship Visualizer*. (Processing can be downloaded from *processing.org/download*.)

Also, it should be acknowleged that the final application incorporates and builds on simple graph drawing code developed by Ben Fry [8]. All development work toward directional edges with quantitative values was done by the author.

4 Results

This section provides a walk-through of the *Relationship Visualizer* application, illustrating some of its potential uses. Mobile phone usage records have been used as the sample data set, with phone numbers obfuscated to protect privacy.

4.1 Data Input

To run successfully, a plain text file named "data.csv" must be placed in the sketch's data folder. The data file must contain edges in the form:

from_node_name, *to_node_name*, *edge_value*

Node names can be alpha or numeric characters, and should not be surrounded by quotation marks. Edge values must be numeric. All values must be comma-separated.

4.2 Default Visualization

When the application is first run, the default, "simple graph"-like visualization is shown (see figure 1). Nodes are represented by gray circles, and node names are shown when the circle diameter is wide enough to accommodate the text label. The radius of each circle reflects the number of connecting edges, so nodes with more connections are shown as larger circles.

Edges are aggregated and represented as simple gray lines between nodes. At this point, neither the multiplicity of edges, nor their directions or associated values are shown.

Although nodes are initially placed using a basic force-directed layout method, they can be dragged with the mouse, and positioned as desired by the user.

Pressing the "L" key will switch to a clustered layout method in which large-value nodes are weighted more toward the center, and low-value nodes are pushed toward the outside.

When using the clustered layout, spacing between nodes can be adjusted using the "J" and "K" keys.

Initially, all nodes are shown. Low-value (small) nodes can be filtered out and later revealed by using the bracket keys.

A basic help screen describing all of this appears (not shown here), and may be toggled off and on using the question mark key.

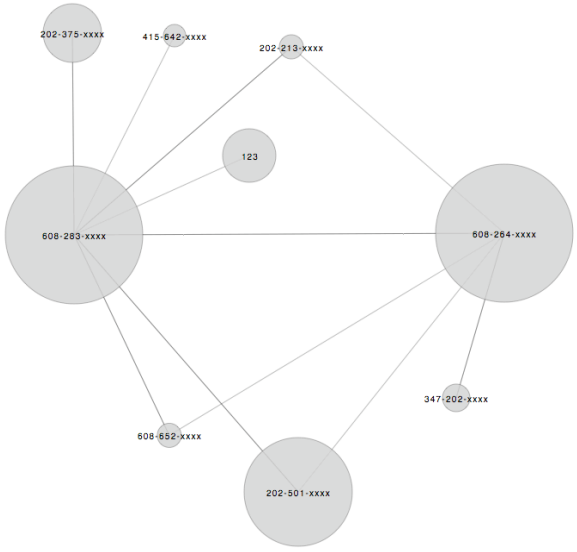

Fig. 1. Default visualization, with many small nodes hidden.

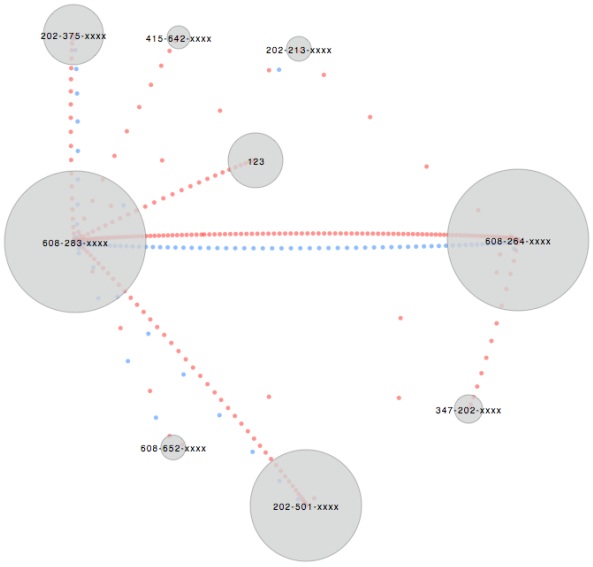

Fig. 2. First visualization with individual edges represented as discrete visual forms.

4.3 Edges Encoded

Pressing "3" reveals the first visualization with edges encoded as discrete visual forms—small blue and red circles, in this case (see figure 2). Motion reveals the edges' directionality, which is also reinforced by the colors. The color choices are arbitrary, and a greater prevalence of blue or red in the visualization should not be considered indicative of any trend; they are simply used here to help distinguish edge direction.

Motion can be accelerated, slowed, or stopped altogether by pressing "S" and "F". Notice how, without lines present, connections with very few edges may be difficult to discern. Increasing the rate of motion makes those low-value connections more easily perceptible, while slowing the rate of motion helps with high-value connections.

With all motion stopped, it may be useful to reveal path guidelines that indicate edge direction by pressing zero (see figure 3). The light gray lines and arrows are particularly useful for generating static visualizations, where motion cannot be used to indicate direction. Pressing "P" exports a copy of the on-screen view as a high-resolution, vector PDF file.

4.4 Quantitative Edge Values Encoded

Pressing "4" reveals an equivalent visualization, but with edge values (in this case, telephone call duration) encoded as the diameter of each edge-circle (see figure 4). Longer phone calls are larger circles, and shorter ones smaller. Immediately, a wealth of new information is present, and the user can identify imbalances and trends within the network of relationships. Telephone number A may call B more often, but when B calls A, they tend to talk for longer periods. Or, A places few outgoing calls, but receives incoming ones from many different numbers.

4.5 Quantitative Edge Values Encoded (Alternate)

Pressing "5" reveals a similar view, but with edges represented as rectangles, and with edge values encoded as the height of each rectangle (see figure 5). This view is similar to a traditional bar chart representation, and may be more appropriate given the data set.

Also note that the paths traveled by edges between nodes are slight arcs, enhancing visibility of the edges, which would otherwise overlap. The plus and minus keys can be used to increase and decrease the width of the arcs, which may improve readability, depending on the desired node placement (see figures 6–9).

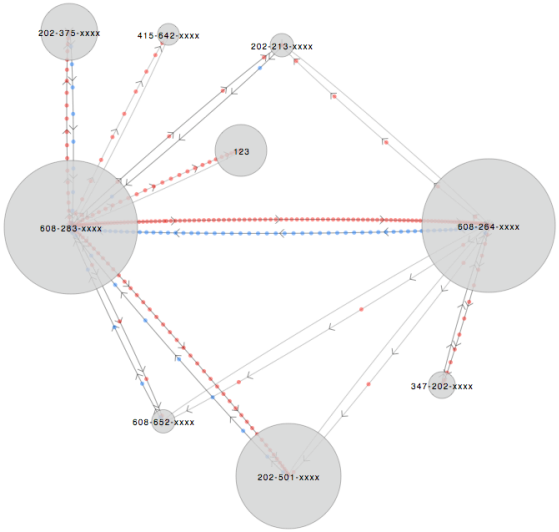

Fig. 3. Discrete edges, with directional guidelines shown.

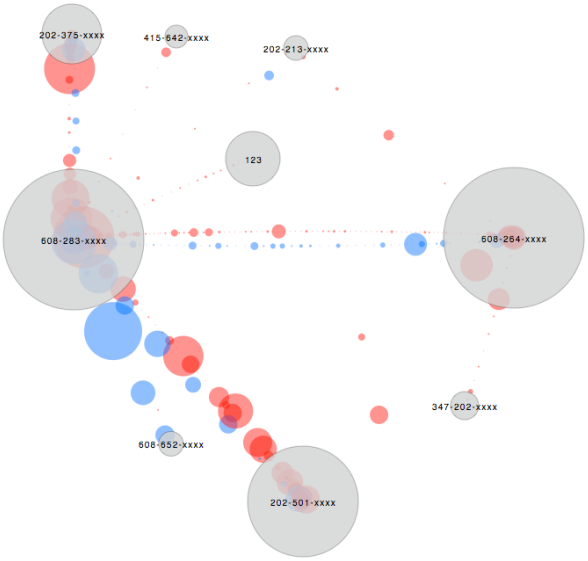

Fig. 4. Discrete edges with quantitative values encoded as circle diameter.

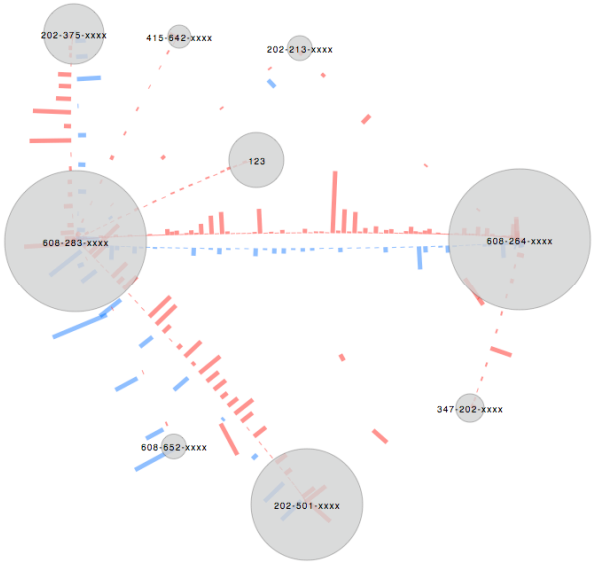

Fig. 5. Discrete edges with quantitative values encoded as bar length.

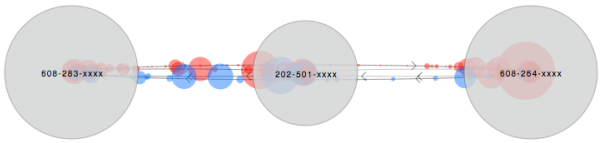

Fig. 6. With nodes arranged along the same axis, and edge paths relatively flat, the edge values are difficult or impossible to discern.

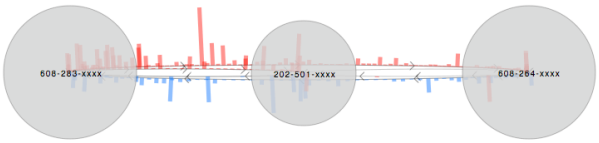

Fig. 7. Even as rectangles, the values are difficult to perceive.

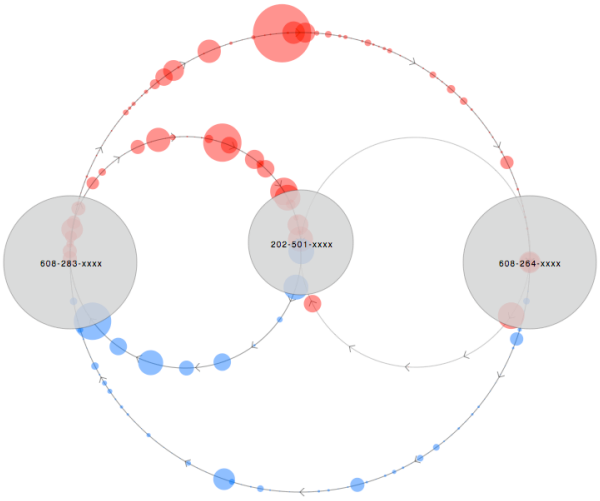

Fig. 8. When the edge path arcs are widened, edge values are much more clearly perceived.

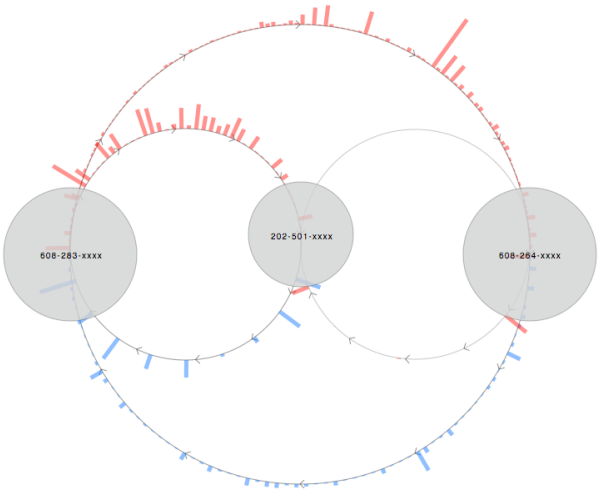

Fig. 9. Same as at left, but with rectangle edges.

5 CONTRIBUTIONS

This project offers a number of contributions toward the future of visualization of directed graphs with multiple edges:

1) The presentation of edges as discrete visual elements
2) The use of motion to indicate edge directionality
3) The encoding of an additional quantitative value for each edge

As discussed in the "Related Work" section, nearly all prior visualizations of graphs with multiple edges maintain the convention of representing the edges as lines. The multiplicity of edges is typically encoded as a visual property of the line, either thickness, brightness, or hue.

The primary contribution of this project is to break with the tradition of representing edges solely as lines. As we can see now, when multiple edges are present, each edge can be visualized discretely, whether as a circle, rectangle, or some other form. Not only is there no need to aggregate multiple edges into singular visual form, but visualizing each edge as its own entity provides opportunities for encoding additional quantitative values specific to each edge. More data can be communicated using discrete edges instead of traditional lines. While lines may be superimposed on this sort of visualization, they are necessary only when few edges are present. Otherwise, the arrangement and motion of edges communicates the binary information originally represented by the simple line: whether or not two nodes are connected. *Connections* are still obvious, yet much more detail about each *relationship* is made visible.

6 CONCLUSION

This research opens the door to a number of other visualization possibilities, such as experimenting with alternate visual forms for edges (beyond just circles and rectangles), incorporating ordering of edges in date/time sequence, encoding additional values onto edges to render additional dimensions of data, and designing legible labeling systems to reveal exact values, where appropriate.

It is the author's hope that others will expand upon the visualization and interaction techniques presented here to present more valuable and data-rich presentations of all kinds of network relationships.

ACKNOWLEDGEMENTS

Thanks to Maneesh Agrawala of UC Berkeley for providing the opportunity to pursue this research, and to Ben Fry for his work on Processing and development of the initial simple graph code from which *Relationship Visualizer* grew. Also, many thanks to Jan Kubasiewicz, Brian Lucid, and Joe Quackenbush of MassArt for their support in my pursuit of novel visualizations of data.

REFERENCES

[1] Y. Jia, J. Hoberock, M. Garland, and J. C. Hart. On the Visualization of Social and other Scale-Free Networks. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1285–1292, 2008.

[2] W. Cui, H. Zhou, H. Qu, P. C. Wong, and X. Li. Geometry-Based Edge Clustering for Graph Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1277–1284, 2008

[3] D. Holten. Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748, 2006.

[4] N. Henry, A. Bezerianos, and J.-D. Fekete. Improving the Readability of Clustered Social Networks using Node Duplication. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1317–1324, 2008

[5] D. Phan, L. Xiao, R. Yeh, P. Hanrahan, and T. Winograd. Flow Map Layout. In *Proc. of IEEE Symposium on Information Visualization (InfoVis '05)*, pages 219–224. IEEE Press, 2003.ber 18-20, 1994, pages pp. 248–261. Springer, 1998.

[6] M. Rosvall and C.T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118, 2008.

[7] E. R. Tufte. *Envisioning Information*. Cheshire, Conn.: Graphics Press, 2006.

[8] B. Fry. *Visualizing Data*. Sebastopol, CA: O'Reilly Media, Inc., 2008.