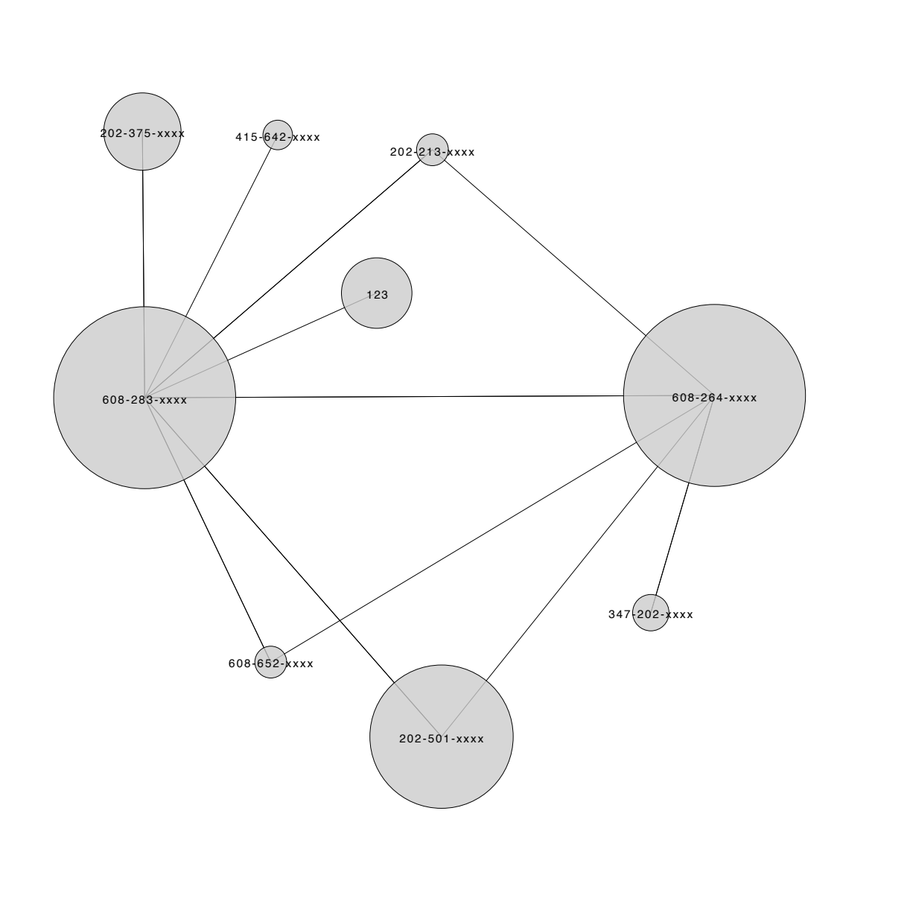
Visualizing network relationships

Scott Murray

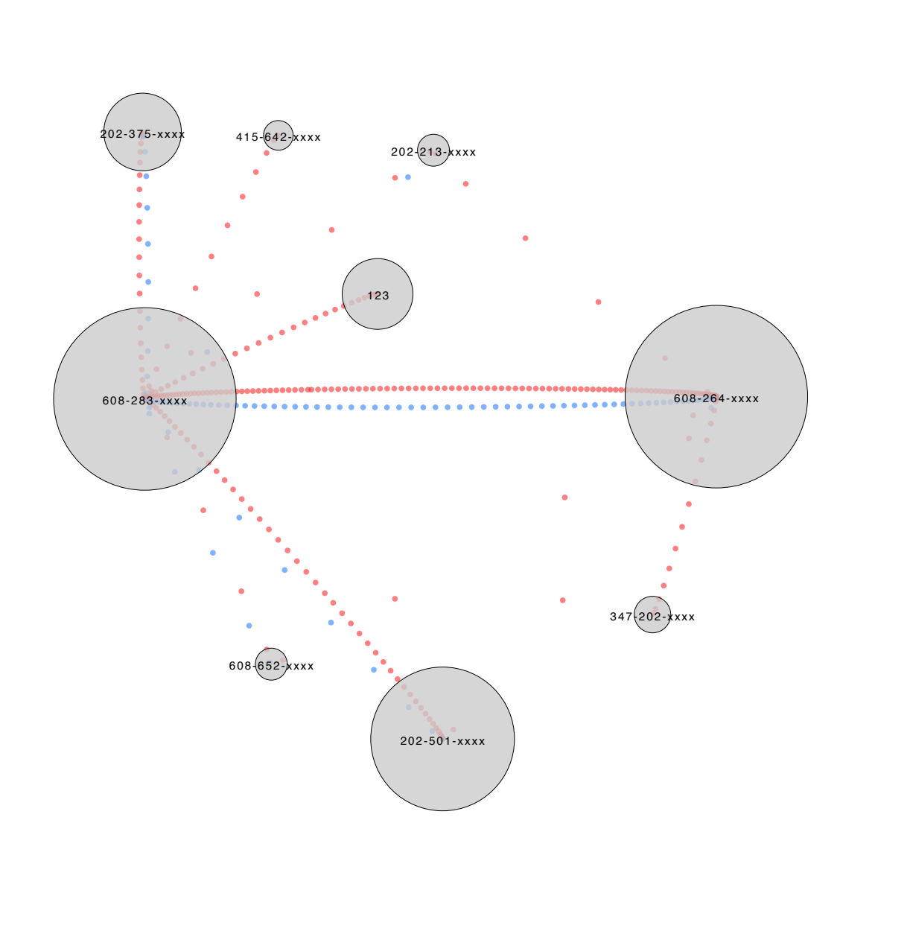
CS294-10
Fall 2008

SIMPLE GRAPH

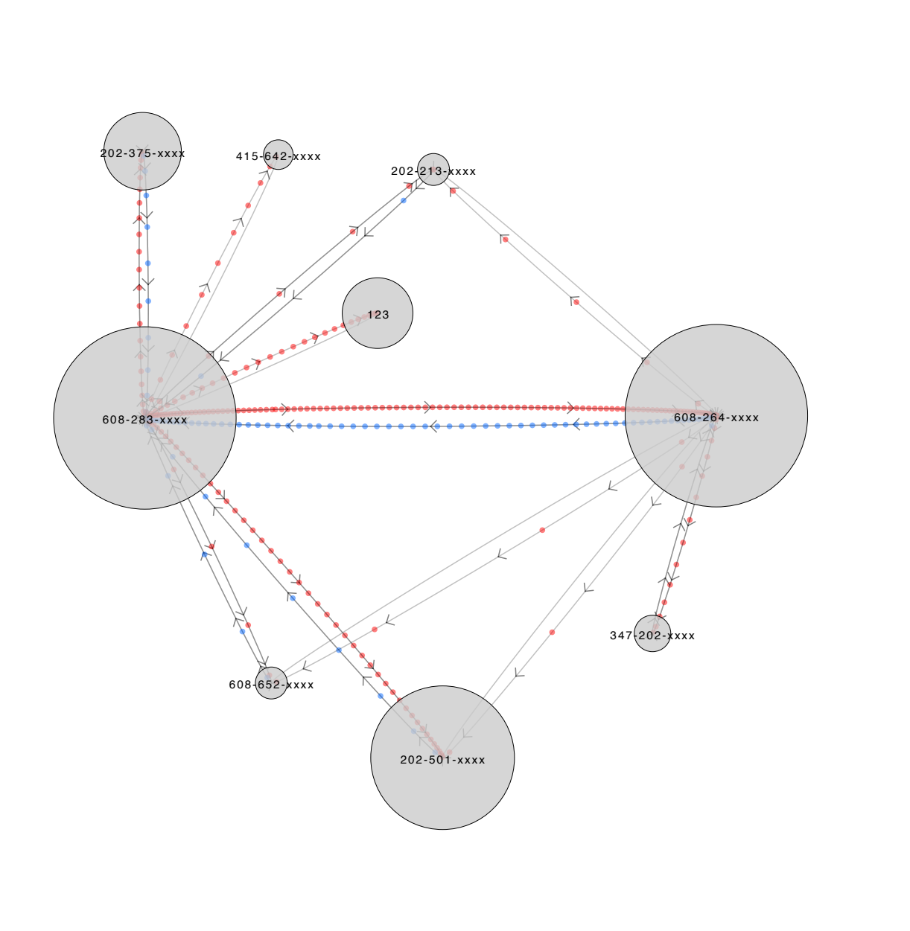


A simple graph rendering. Any multiple edges are obscured.

MULTIPLE EDGES

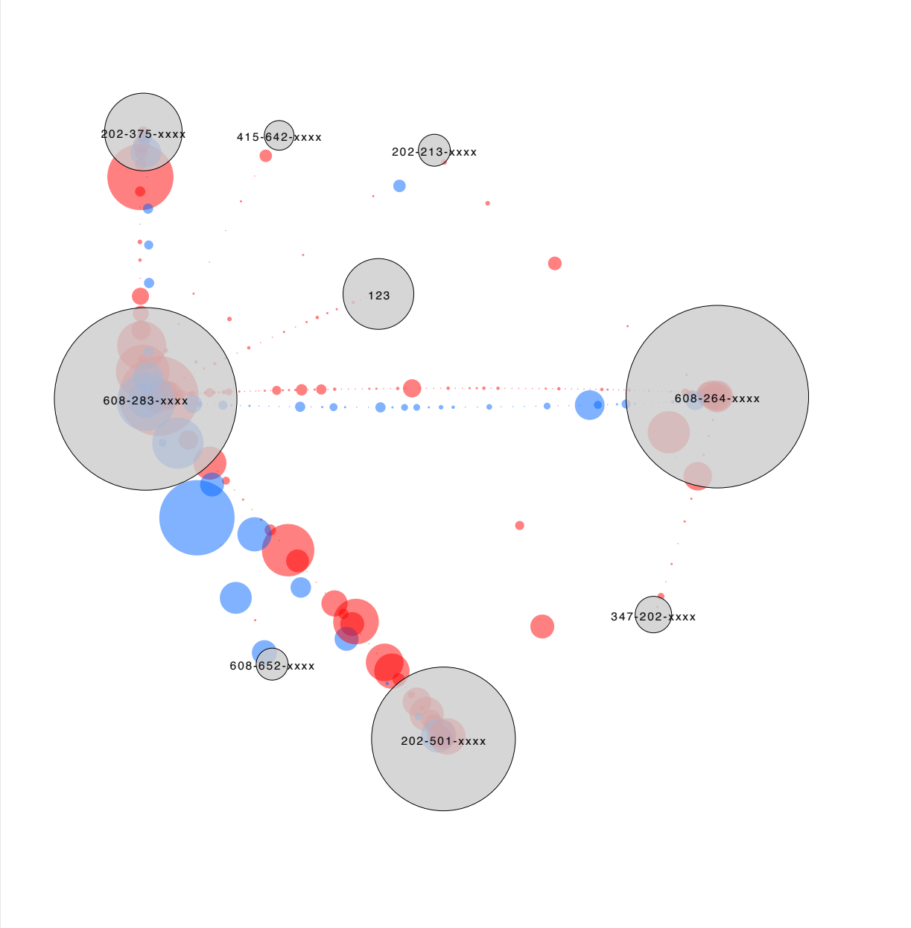


Each red or blue circle represents one edge, or one instance of connection between the nodes. (In this case, one phone call.) On-screen, red and blue edges move in opposite directions.

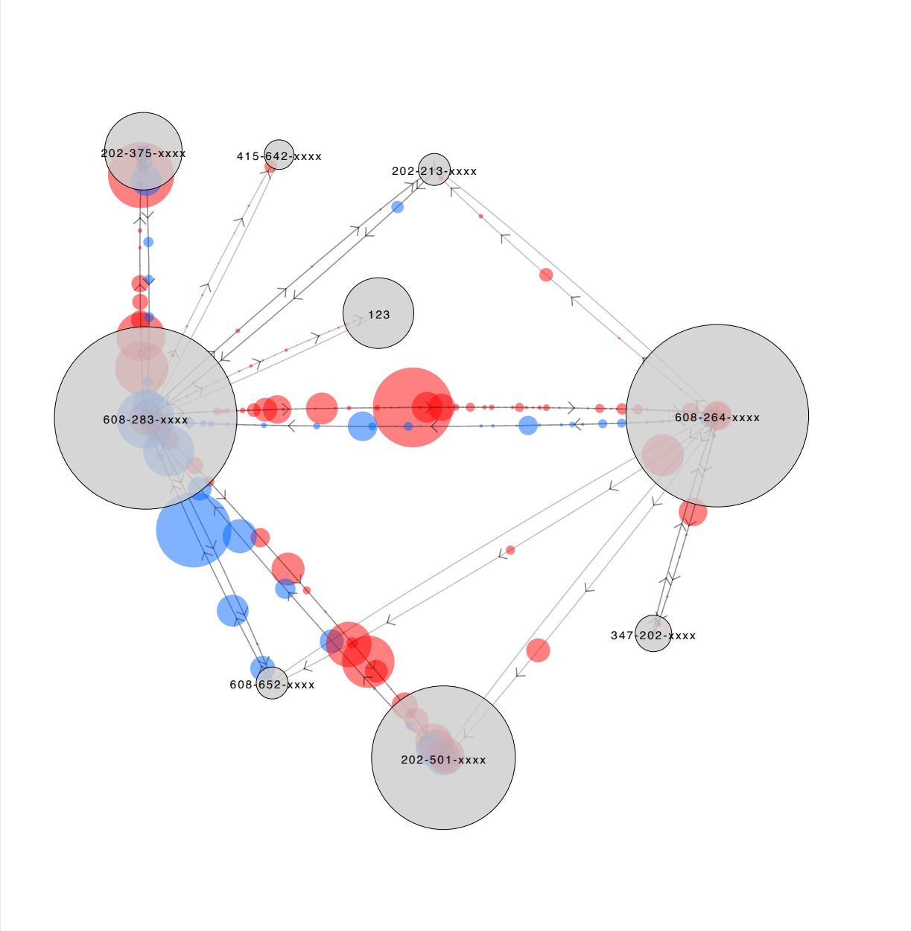


Same as above, with directional guidelines shown.

EDGE VALUES ENCODED

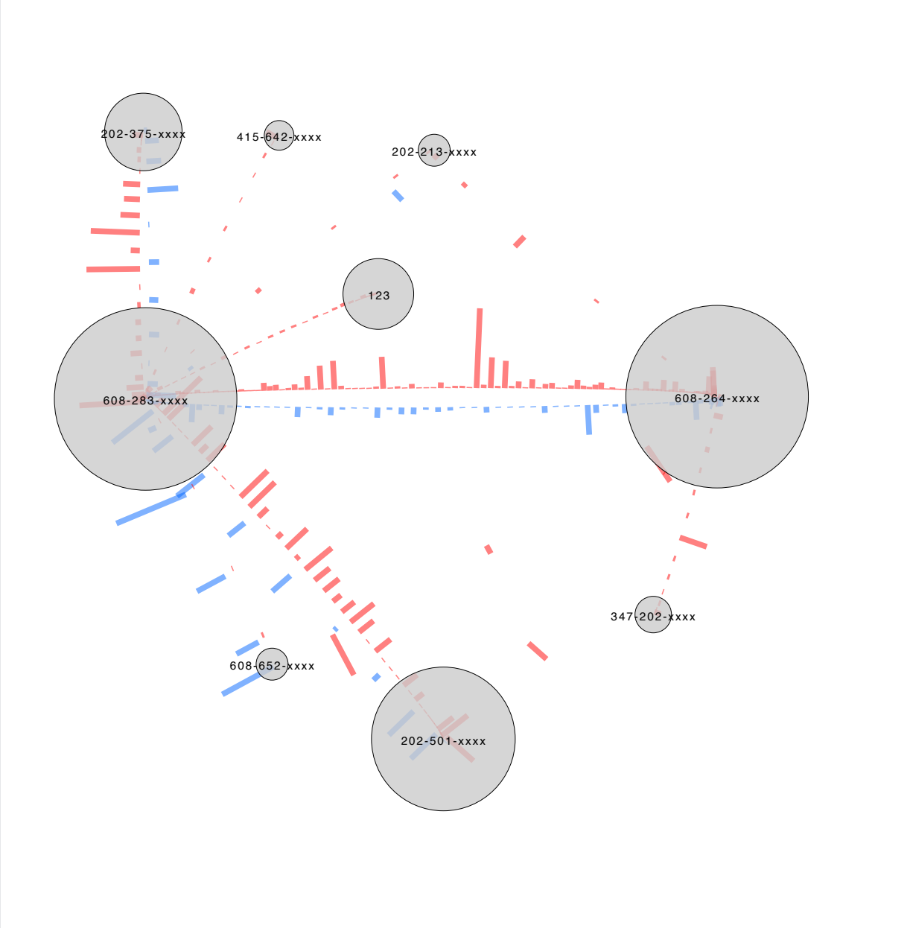


Edge circles are scaled to reflect each edge value (e.g., the length of the phone call in minutes). Patterns emerge—we see who calls whom more frequently, or those who call less often but talk longer on each call.

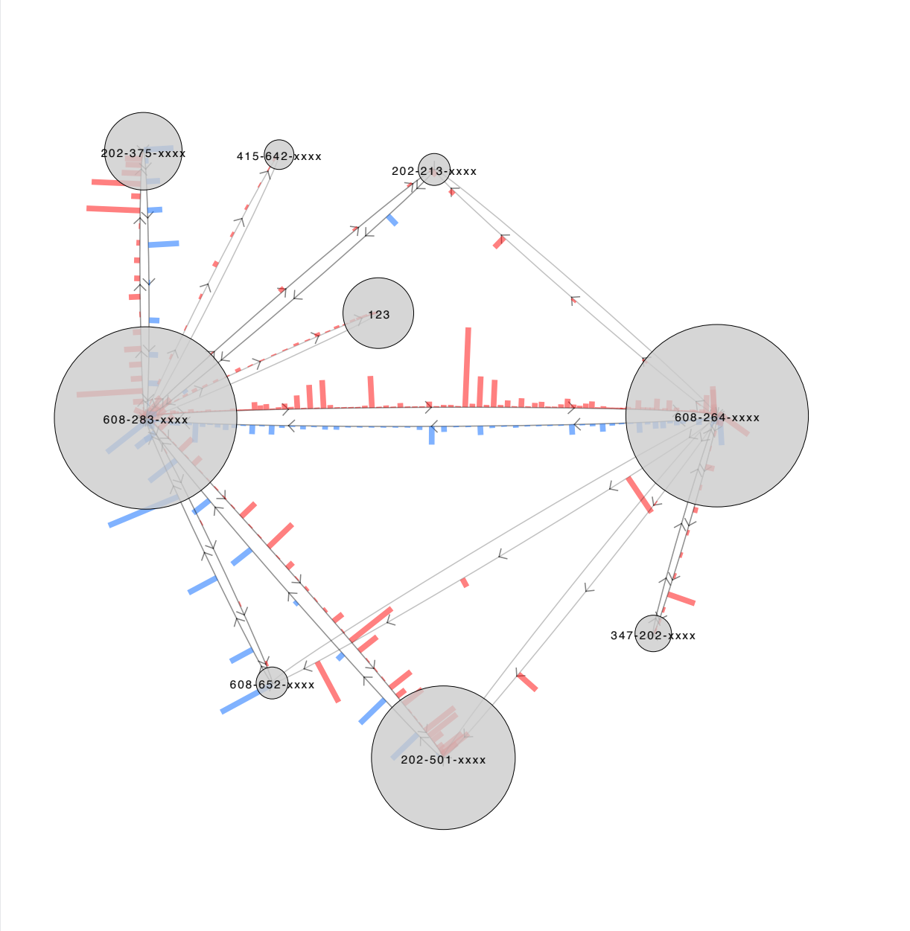


Same as above, with directional guidelines shown.

ALTERNATE ENCODING

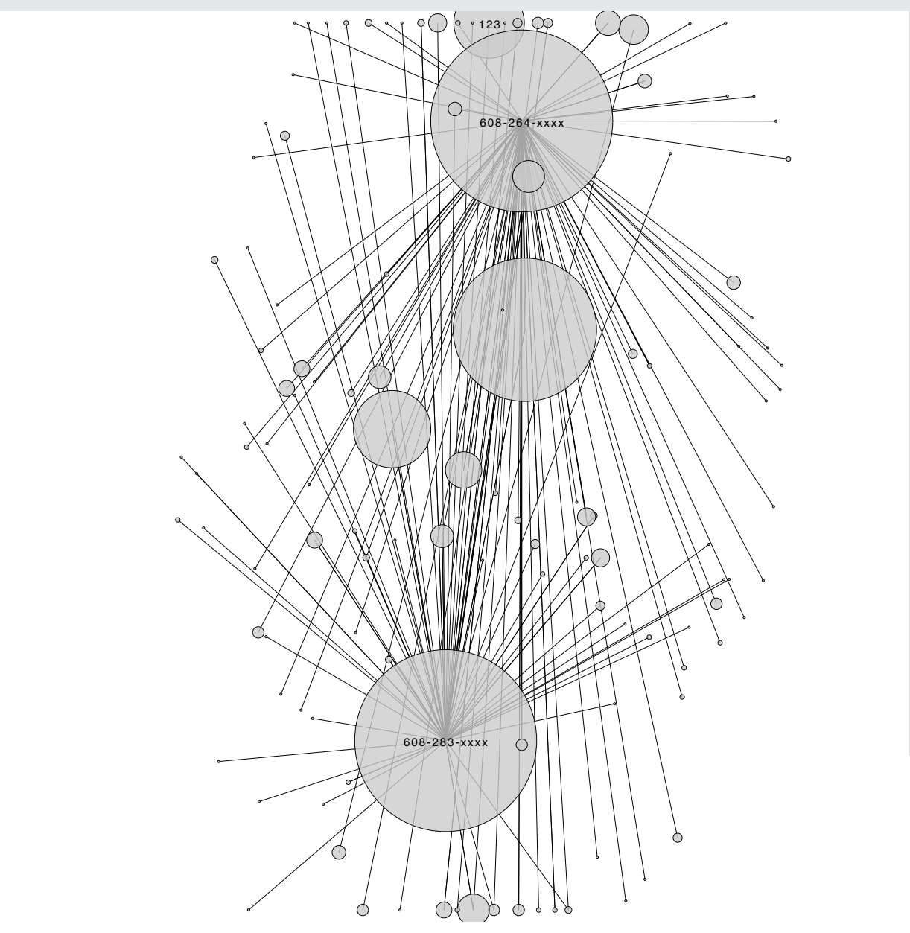


The same edge/phone call values as shown at left, but represented with bar chart-like rectangles. Rectangle lengths are more easily interpreted than changes in circle diameter.

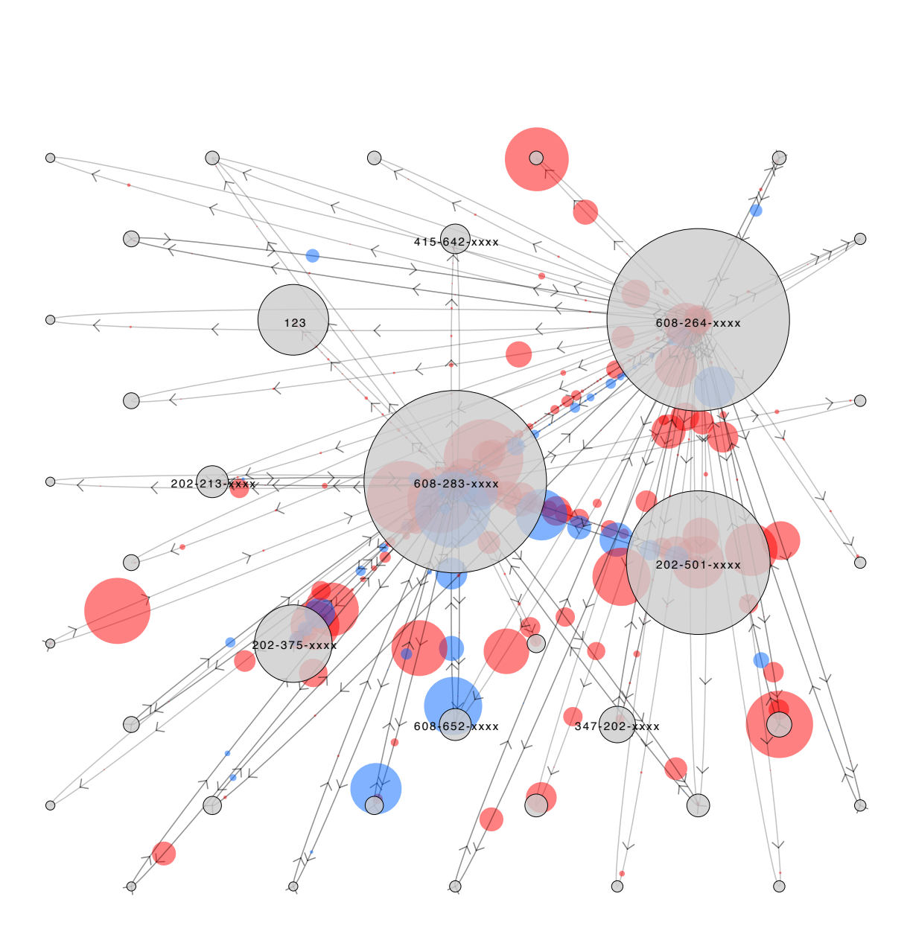


Same as above, with directional guidelines shown.

AUTOMATED LAYOUTS

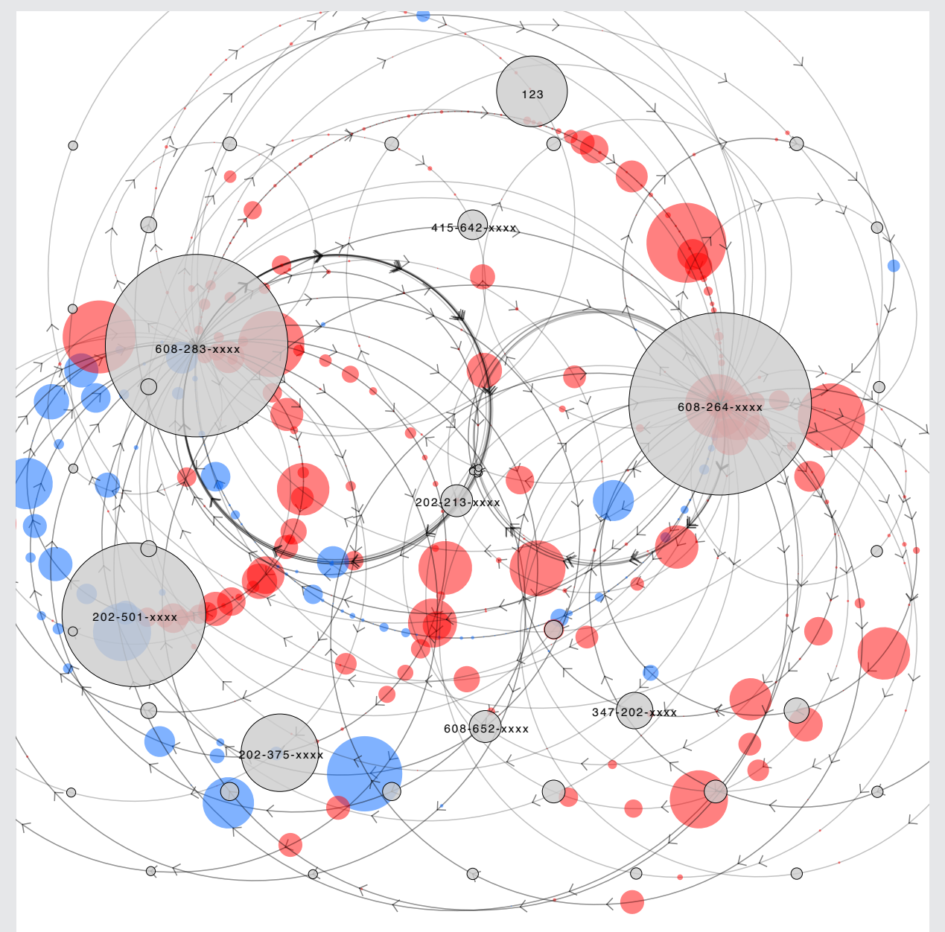


Initial layout using a simplified force-directed technique. Note that the user can override node placement at any time by dragging nodes to the desired positions on the screen.

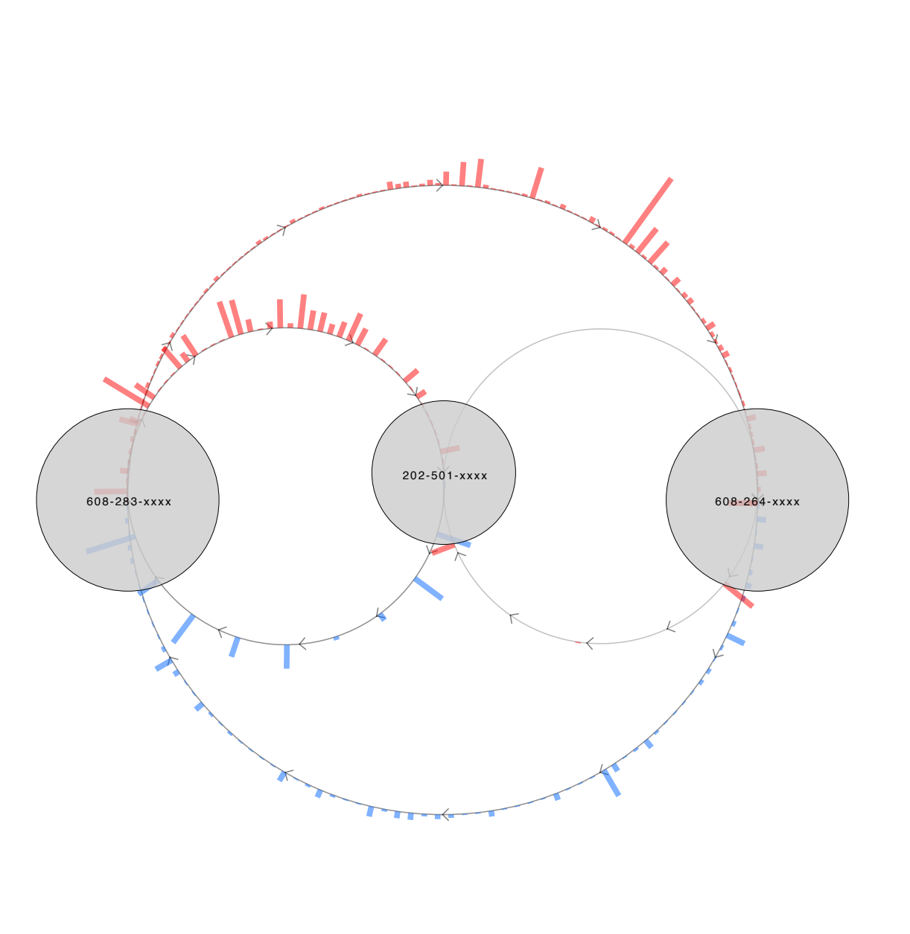


Layout using weighted clustering technique. Note that larger nodes are placed nearer the center. This method better facilitates user exploration of high-value nodes, especially as more low-value nodes are hidden from view.

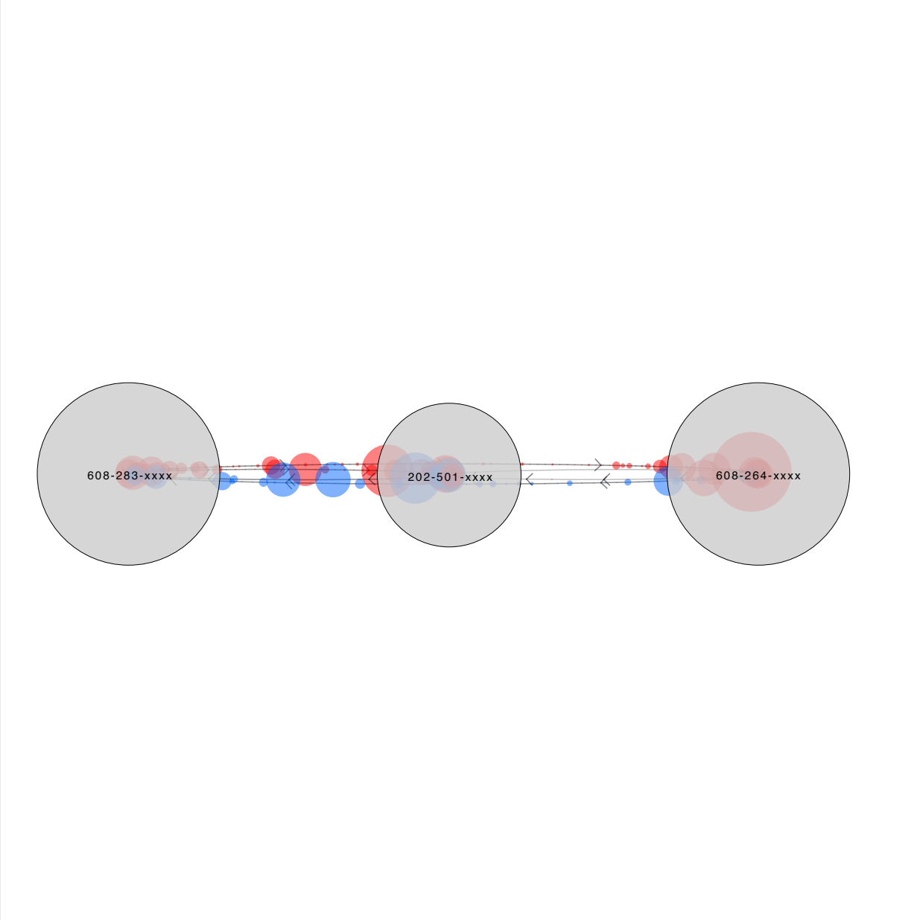
VARIABLE ARC WIDTH



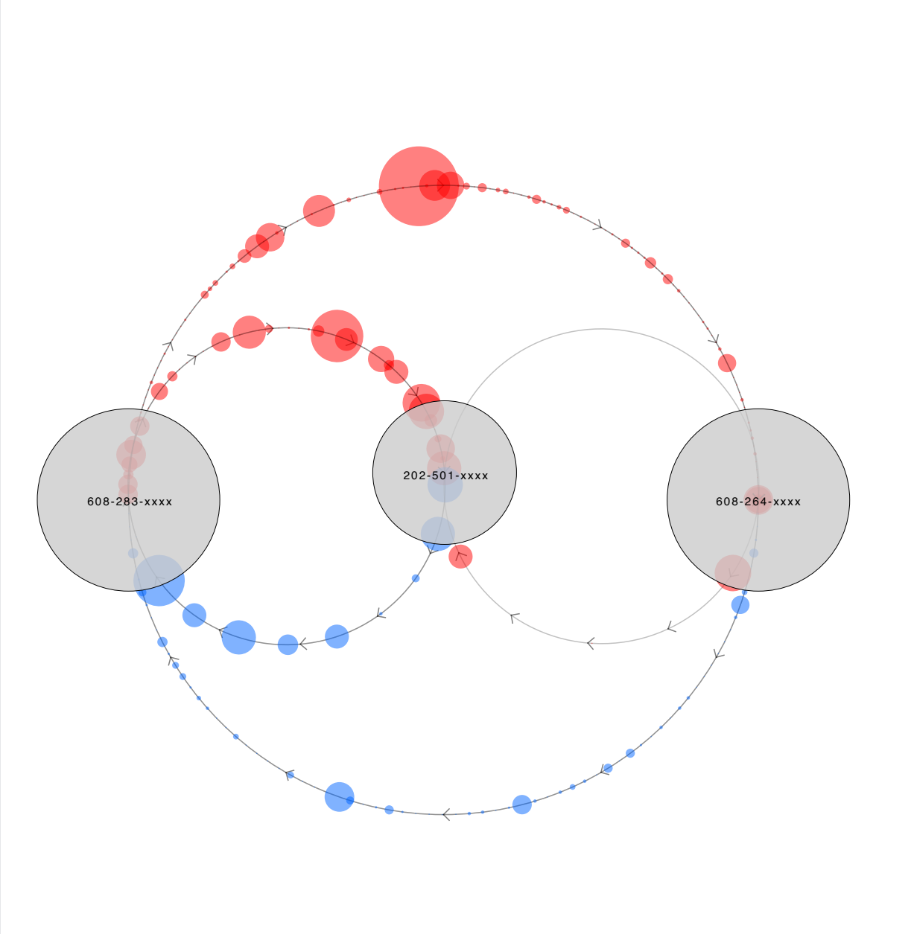
Wider edge arcs may create interesting visual patterns, but are not conducive to interpretation of the data when used with many nodes.



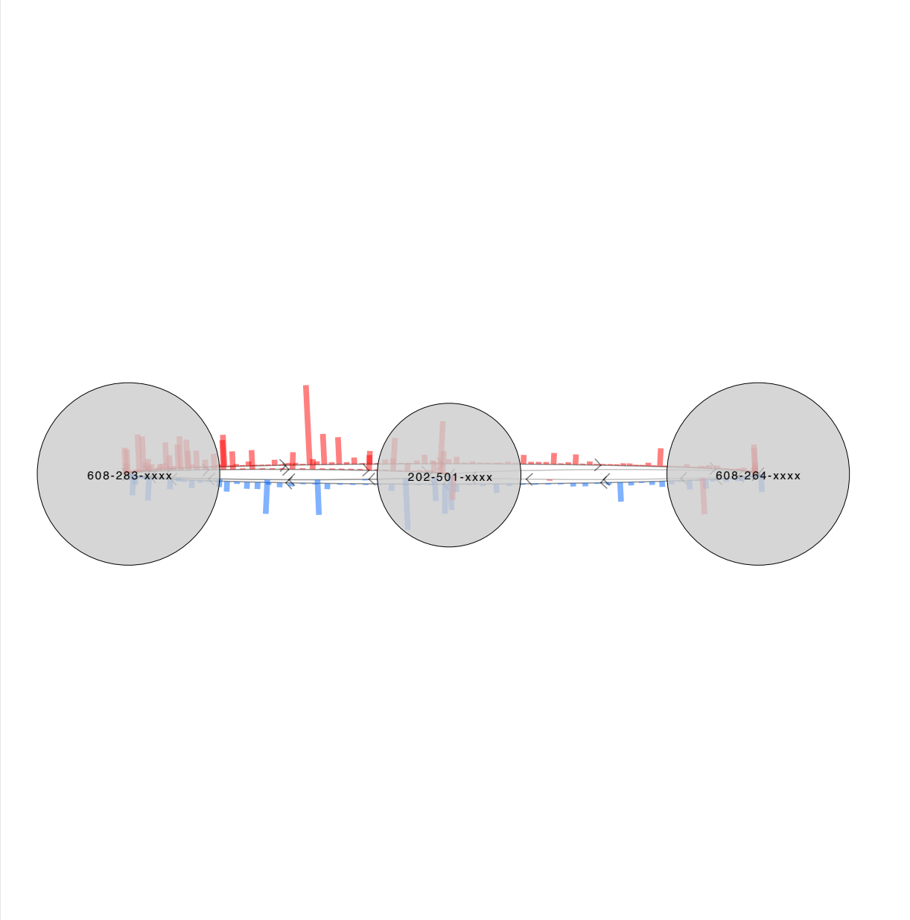
When examining only a few nodes at a time, however, using wide arcs may help clarify the data and enable interesting layouts not possible with straight edge lines (such as this arrangement with one node directly between two larger nodes).



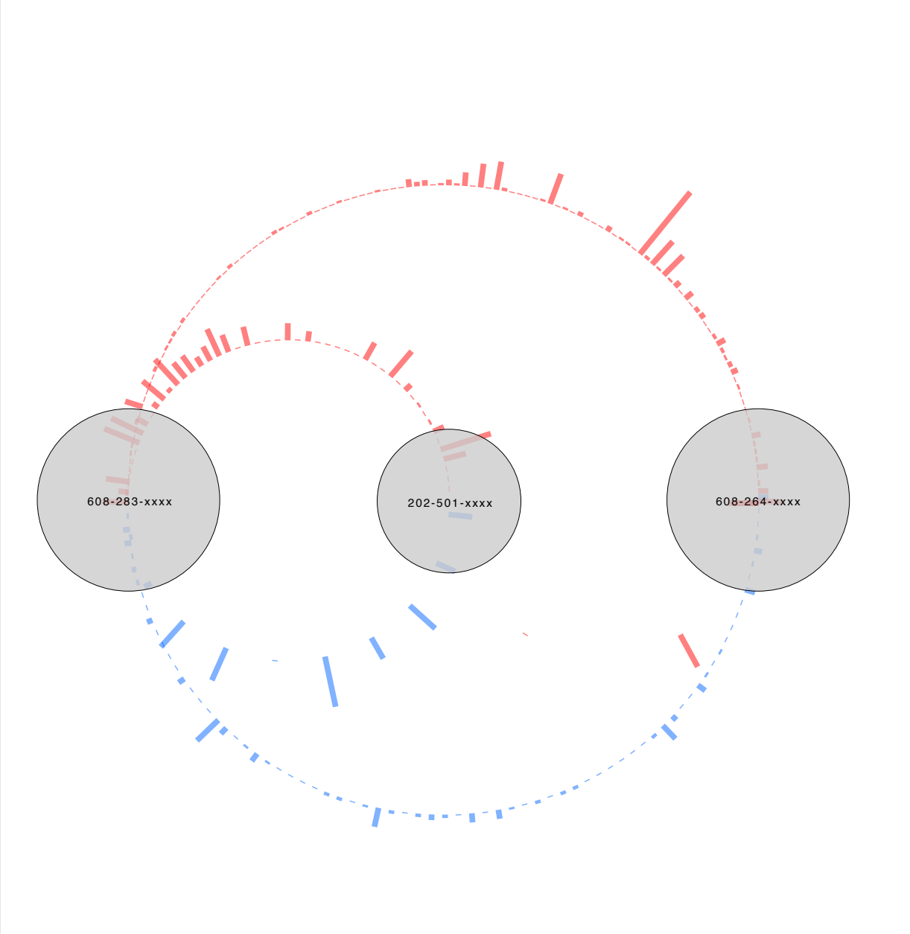
With a very low arc, these edges overlap, obscuring each other. Interpreting this rendering is difficult or impossible.



With full, circular arcs, the edges are much more clear, and the relationships between each of the three nodes is readily visible.



For comparison, the same nodes and data as shown at left, but with edge values encodes as rectangles.



Again, the same nodes and data as shown at left, but with edge values encodes as rectangles, and directional guidelines hidden. Note that, with this rectangle encoding, the edge paths are implicit, yet perceptible when there are many edge values present.

PROJECT OVERVIEW

Problem

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, the focus of this project.

Motivation

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 *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, social networks, economic trade data, website links, and network traffic.

Approach

I began the project with the phone records of two individuals (the primary nodes), including the length of each phone call in minutes (the edge values). I then experimented with ways to encode the edges with multiple values for each node-to-node relationship, starting first with simply the number of phone calls and the direction of the call (e.g., from A to B, or from B to A). Then I incorporated motion to more intuitively convey the directionality of calls, as well as a scale for each call, so that the length of each phone call (the edge's quantitative value) was represented.

Since my intent was to build a tool that could be used with a wide range of data sets, I used a simplified data model. The application takes an input of a plain text file with each line representing one edge, in the format:

from_node_name, to_node_name, quantitative_value

Recognizing that no single visual representation could be ideal for all possible data sets, I incorporated some basic interaction tools to enable users to filter the data, modify visual properties, and focus on what is most relevant to them. PDF export functionality allows users to easily save a static visualization for later review.

Results

"Relationship Visualizer" is an interactive application, built with Processing, that encourages users to input their own graph data and manipulate the visualization, modifying parameters until arriving at an rendering appropriate for the user's needs.

Future Work

—Incorporate additional data dimensions, such as date/time values for edges, and design methods of accommodating each additional layer of data into the multivariate display.
—Implement alternate layout methods.
—Incorporate mouse rollover behavior to reveal information on selected node and/or connected nodes.
—Factor in automatic scaling, so minimum/maximum edge values will be scaled appropriately for the display resolution.
—Build a more intuitive user interface, less reliant on keyboard commands.
—Make more parameters (e.g. color, line weight, opacity) user-modifiable.
—Solicit user feedback and incorporate additional visualization methods.